MASTER EN SCIENCES INFORMATIQUES

Facebook Connect, OpenID, OAuth, Oath... Une jungle de protocoles ou différents noms pour une même réalité ?

Vermeren, Vincent

Award date:
2012

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix, Namur

Faculté d'Informatique

Année académique 2011-2012

Facebook Connect, OpenID, OAuth, OAth... Une jungle de protocoles ou différents noms pour une même réalité?

Vincent Vermeren

Mémoire présenté en vue de l'obtention du grade de master en Sciences Informatiques

Résumé

La gestion de l'identité en informatique est un domaine très large. Il est difficile de classer un système existant dans une catégorie précise. Les buts de ce mémoire sont une étude et une analyse comparative de plusieurs systèmes d'authentification unique destinés à un environnement web. Ces systèmes sont basés sur des standards ou sont des modules de gestion de l'authentification à communauté large. Les aspects « autorisation » et « délégation de l'autorisation » sont également abordés.

L'étude porte principalement sur le flux d'authentification et d'autorisation de chacun des modules, leur sécurité et l'aspect « respect de la vie privée » pour chacun de ceux-ci.

L'analyse comparative de ces différents systèmes propose une méthodologie pour aider à la sélection d'un système au cas par cas. Les critères de choix repris dans les différents tableaux sont les aspects authentification, autorisation, sécurité et vie privée étudiés dans ce mémoire.

Mots-clés: systèmes de fédération d'identité, authentification unique, délégation des autorisations, systèmes de gestion de l'identité, accès aux ressources protégées.

Abstract

Identity management in computer sciences is a very wide domain. It's difficult to classify an existing system in a specific category. The goals of this master's thesis are a study and comparative analysis of several single sign-on systems for web environment. Those systems are based on standards or they are authentication management modules for a wide community. The « authorization » and « delegated authorization » aspects are also discussed.

The study will focus on the authentication and authorization flows of each module, their security and their privacy aspects.

The comparative analysis of these systems provides a methodology to help the selection of a system on a case-by-case basis. The selection criteria included in the tables are the aspects authentication, authorizations, security and privacy studied in this master's thesis.

Keywords: identity federation systems, Single Sign-On, delegated authorizations, identity management system, access to protected resources.

Avant-propos

Je souhaite adresser mes remerciements les plus sincères aux personnes qui m'ont, de près ou de loin, aidé à l'élaboration de ce mémoire.

Je tiens, tout d'abord et particulièrement, à exprimer ma reconnaissance et ma gratitude au Professeur Jean-Noël Colin, promoteur du mémoire, pour ses conseils avisés, sa disponibilité et le temps qu'il a pu m'accorder pendant cette année universitaire.

Je remercie Benjamine Lurquin pour sa disponibilité et les renseignements fournis tout au long de ce cursus scolaire.

Mes remerciements s'adressent également à Geoffroy Simon, pour ses relectures et ses conseils expérimentés.

J'adresse mes plus sincères remerciements à mes proches et mes amis qui m'ont toujours soutenu et encouragé.

Je remercie également mes parents et en particulier mon père pour ses encouragements à suivre des études universitaires.

Enfin, je remercie particulièrement ma compagne, Elia, pour son soutien tout au long de mes études.

Table des matières

Avant-propos i

Table des matières iii

Table des figures vi

Table des tableaux vii

Glossaire viii

1 Introduction 1

2 Etat de l'art 2
- **2.1 Préambule** 2
- **2.2 Authentification vs identification** 2
- **2.3 Autorisation et délégation de l'autorisation** 2
- **2.4 Gestion de l'identité** 3
- **2.5 Authentification unique** 5
- **2.6 Déconnexion unique** 6
- **2.7 Types de document** 6
- **2.8 Types d'attaque** 7

3 OAuth 9
- **3.1 Introduction** 9
- **3.2 OAuth v1.0a** 9
  - 3.2.1 Type et composition 9
  - 3.2.2 Concepts de base 10
  - 3.2.3 Etude du protocole 10
  - 3.2.4 Considérations en termes de sécurité 13
  - 3.2.5 Failles de sécurité 14
  - 3.2.6 Synthèse 14
- **3.3 OAuth v2** 15
  - 3.3.1 Type et composition 15
  - 3.3.2 Les nouveaux concepts 15
  - 3.3.3 Concepts de base 16
  - 3.3.4 Etude du protocole 17
  - 3.3.5 Considérations en termes de sécurité 24
  - 3.3.6 Failles de sécurité 27
  - 3.3.7 Synthèse 28

4 Facebook Connect 29
- **4.1 Introduction** 29
- **4.2 Facebook Connect** 29
- **4.3 Type et composition** 31
- **4.4 Concepts de base** 31
- **4.5 Etude du protocole** 32
  - 4.5.1 Flux coté serveur 32
  - 4.5.2 Flux coté client 34
- **4.6 Réflexion sur les acteurs repris dans l'état de l'art** 36
- **4.7 Considérations en termes de sécurité** 36
  - 4.7.1 Remarques communes aux deux flux 36
  - 4.7.2 Sécurité du flux coté serveur 37
  - 4.7.3 Sécurité du flux coté client 37

4.7.4 Sécurité de l'accès aux ressources 37
4.7.5 Résistance aux attaques 37
4.8 Respect de la vie privée 38
4.9 Critiques diverses 41
4.10 Synthèse 41

5 OpenID 2.0 42
5.1 Introduction 42
5.2 Type et composition 43
5.3 Concepts de base 44
5.4 Etude du protocole 45
5.4.1 Flux de communication 45
5.4.2 Flux d'authentification 45
5.4.3 Découverte des tiers OpenID 48
5.4.4 Extensions du protocole OpenID v2 48
5.5 Considérations en termes de sécurité 48
5.5.1 Prévention des attaques 48
5.5.2 Failles de sécurité 51
5.6 Respect de la vie privée 51
5.7 Synthèse 52

6 OpenID Connect 1.0 54
6.1 Introduction 54
6.2 Type et composition 54
6.3 Concepts de base 57
6.4 Etude du protocole 59
6.4.1 Profil du client de base 59
6.4.2 Découverte 61
6.4.3 Inscription dynamique 62
6.5 Considérations en termes de sécurité 63
6.6 Respect de la vie privée 66
6.7 Synthèse 67

7 SAML 2.0 68
7.1 Introduction 68
7.2 Type et composition 69
7.3 Concepts de base 70
7.3.1 Assertions 70
7.3.2 Protocoles 71
7.3.3 Liaisons 71
7.3.4 Profils 72
7.3.5 Metadata 72
7.3.6 Contexte d'authentification 73
7.3.7 Liens entre les différents composants 73
7.4 Etude du protocole 73
7.4.1 Fédération d'identité 74
7.4.2 Profil d'authentification unique via navigateur web 74
7.4.3 Profil de découverte des fournisseurs d'identité 76
7.4.4 Profil de déconnexion unique 77
7.5 Considérations en termes de sécurité 78
7.6 Respect de la vie privée 78
7.7 Synthèse 79

8 Grilles de comparaison 80
8.1 Préambule 80
8.2 Tableaux comparatifs 81

8.2.1 Qualité de la documentation 81
8.2.2 Richesse des données 82
8.2.3 Fonctionnalités et services 83
8.2.4 Scénarios couverts 85
8.2.5 Contre-mesures face aux menaces 86
8.2.6 Respect de la vie privée 89
8.2.7 Interopérabilité 92
8.2.8 Facilité d'implémentation 94
8.2.9 Maturité 95
8.3 Tableau récapitulatif 96
8.4 Choix d'un système 96
8.5 Synthèse 98

9 Conclusion 100

10 Bibliographie 102

11 Annexes 108
11.1 Les autres systèmes 108
11.2 OAuth v1.0a : le flux d'authentification et ses paramètres 109
11.3 Facebook Connect : configuration de l'application Facebook 110
11.4 OpenID 2.0 : choisir son fournisseur 112
11.5 OpenID Connect 1.0 : spécifications complémentaires 113
11.5.1 Gestion de session 113
11.5.2 Standard 116

Table des figures

Figure 0-1 : Composition d'un URI [Wikipedia, 2012b] ix
Figure 2-1 : Acteurs de la gestion de l'identité et leurs relations [Bertino et Takahashi, 2011] 4
Figure 3-1 : Flux d'authentification OAuth v1.0a [Trenka, 2009] 11
Figure 3-2 : Flux abstrait du protocole OAuth v2.0 16
Figure 3-3 : Flux du code d'autorisation OAuth v2.0 19
Figure 3-4 : Flux d'octroi implicite OAuth v2.0 21
Figure 3-5 : Flux des identifiants par mot de passe du propriétaire de la ressource OAuth v2.0 22
Figure 3-6 : Flux identifiants du client OAuth v2.0 23
Figure 3-7 : Actualiser un jeton d'accès expiré OAuth v2.0 23
Figure 4-1 : Données et services de la plate-forme Facebook [Nam Ko, 2010] 29
Figure 4-2 : Accès à la plate-forme Facebook via l'interface API [Tamada, 2011] 31
Figure 4-3 : Flux coté serveur de Facebook Connect [Facebook, 2011b] 32
Figure 4-4 : Boîtes de dialogue OAuth de Facebook Connect 33
Figure 4-5 : Flux coté client de Facebook Connect [Facebook, 2011b] 35
Figure 4-6 : Panneau de configuration Facebook des informations transmises par nos amis 39
Figure 4-7 : Tentative de connexion Facebook après modification du mot de passe 39
Figure 5-1 : Flux abstrait du protocole OpenID 2.0 [Walther, 2011] 44
Figure 5-2 : Flux d'authentification OpenID 2.0 46
Figure 6-1 : Ensemble de spécifications OpenID Connect 1.0 [Dingle, 2011] 55
Figure 6-2 : Flux abstrait OpenID Connect 1.0 58
Figure 6-3 : Flux d'octroi implicite OpenID Connect 1.0 (profil du client de base) 60
Figure 7-1 : Ensemble de spécifications et extensions de SAML 2.0 [Ragouzis et al, 2008] 69
Figure 7-2 : Schéma de l'imbrication des concepts de base de SAML 2.0 [Ragouzis et al, 2008] 70
Figure 7-3 : Relations entre les composants SAML 2.0 [Ragouzis et al, 2008] 73
Figure 7-4 : Cycle d'une session d'authentification unique SAML 2.0 [Bertino et Takahashi, 2011] 74
Figure 7-5 : Flux SAML 2.0 initié par le fournisseur de services avec liaisons redirection/POST 75
Figure 7-6 : Flux SAML 2.0 initié par le fournisseur d'identité avec liaison POST 76
Figure 7-7 : Déconnexion distribuée SAML 2.0 77
Figure 11-1 : Flux d'authentifcation OAuth v1.0a et les paramètres correspondants [Fiat, 2009] 109
Figure 11-2 : Création d'une nouvelle application sur Facebook Connect 110
Figure 11-3 : Insertion du captcha de sécurité Facebook Connect 110
Figure 11-4 : Validation du compte Facebook Developers par SMS 110
Figure 11-5 : Sécurité de Facebook Developers 111
Figure 11-6 : Panneau de configuration Facebook Developers 111
Figure 11-7 : Analyse comparative de différents fournisseurs OpenID 112
Figure 11-8 : Utilisation du flux d'octroi implicite OpenID Connect 1.0 (jeton ID) 113
Figure 11-9 : Flux par code d'autorisation OpenID Connect 1.0 (jeton ID) 114
Figure 11-10 : Flux générique OpenID Connect 1.0 : application native avec un 4ème intervenant 114
Figure 11-11 : Jeton ID synchronisé entre l'application native et le user-agent 115

Table des tableaux

Table 1 : Qualité de la documentation 82
Table 2 : Richesse des données 83
Table 3 : Fonctionnalités et services 84
Table 4 : Scénarios couverts 86
Table 5 : Contre-mesures face aux menaces 88
Table 6 : Respect de la vie privée 91
Table 7 : Interopérabilité 93
Table 8 : Facilité d'implémentation 94
Table 9 : Maturité 95
Table 10 : Aperçu global des systèmes 96
Table 11 : Sélection et pondération des critères 97
Table 12 : Application des critères pondérés aux systèmes 97

Glossaire

Le glossaire informatique repris ci-dessous est applicable dans le cadre de ce mémoire mais peut avoir une signification quelque peu différente en dehors de celui-ci. Ces définitions sont principalement tirées du site internet Wikipédia et de la littérature référencée dans la bibliographie.

API (Application Programming Interface) : interface fournie par un programme informatique permettant d'interagir avec d'autres programmes ou d'autres API. Elle est indispensable pour l'interopérabilité entre les composants logiciels.

Assertion : proposition donnée et soutenue comme vraie.

Captcha : test permettant de différencier de manière automatisée un utilisateur humain d'un ordinateur. Il est souvent représenté sous forme d'une image possédant des caractères que l'utilisateur doit introduire dans un champ de texte.

Complétion : action de compléter, de façon automatique, la saisie au clavier. Aide à la saisie d'information.

Endpoint : point de terminaison pouvant appeler ou être appelé. Il peut générer ou terminer un flux d'information. Il s'agit principalement d'un point d'entrée vers un service, un processus,…

Hash : chaine de caractères produite par application successive de fonctions cryptographiques sur une chaine de caractères.

Horodatage (timestamping) : mécanisme qui consiste à associer une date et une heure à un événement, une information ou une donnée informatique.

Identifiant client : chaîne de caractères unique au sein du système et représentant l'information d'enregistrement fournie par le client et exposée au propriétaire de la ressource. En d'autres termes, il pourrait s'agir de l'identifiant de l'utilisateur. Il peut également être représenté sous forme d'un URI.

Implémentation : mise en œuvre à partir d'un document de conception, d'une spécification, d'un cahier de charge, d'un standard,…

Interopérabilité : capacité que possède un système à fonctionner avec d'autres systèmes sans restriction d'accès ou de mise en œuvre.

JSON (Javascript Object Notation): format de données textuel, générique, dérivé de la notation des objets du langage ECMAScript (langage de programmation de type script) permettant de représenter de l'information de manière structurée [Crockford, 2011].

MAC (Media Access Control) est un identifiant physique stocké dans une carte réseau.

MAC (Message Authentication Code): code accompagnant des données dans le but, par exemple, de s'assurer de l'intégrité de ces dernières après une transmission. Ce code se base non seulement sur le message en lui-même mais également sur une clé secrète.

Namespace : lieu abstrait conçu pour accueillir des ensembles de termes appartenant à un même répertoire.

Nonce : nombre aléatoire utilisé pour signer un ensemble de données d'une communication électronique. Il permet notamment d'éviter les attaques par rejeu.

OIDF (OpenID Foundation): sigle de la fondation OpenID. Elle s'occupe de promouvoir, protéger, et alimenter la communauté et les technologies OpenID.

PKI (Public Key Infrastructure) ou infrastructure à clé publique : ensemble de composants physiques, procédures humaines et logicielles gérant le cycle de vie des certificats numériques/électroniques.

Plugin ou plug-in: petit logiciel qui se greffe à un programme principal pour lui apporter de nouvelles fonctionnalités.

Principal : entité étant la cible d'une requête et pouvant être authentifiée.

REST (REpresentational State Transfer) : style architectural original du Web où un composant lit ou modifie une ressource en utilisant une représentation de celle-ci.

RESTFul : qui suit les principes REST.

SOAP (Simple Object Access Protocol) : protocole permettant la transition de messages entre objets distants.

SWD (Simple Web Discovery) : mécanisme basé sur des requêtes HTTPS GET permettant de découvrir l'endroit d'un type donné de service sur base du nom de domaine.

URI (Uniform Resource Identifier) : courte chaine de caractères identifiant une ressource physique ou abstraite sur un réseau et dont la syntaxe respecte la norme RFC3986 [Berners-Lee et al, 2005]. Une synthèse de la composition d'un URI est représentée à la Figure 0-1.

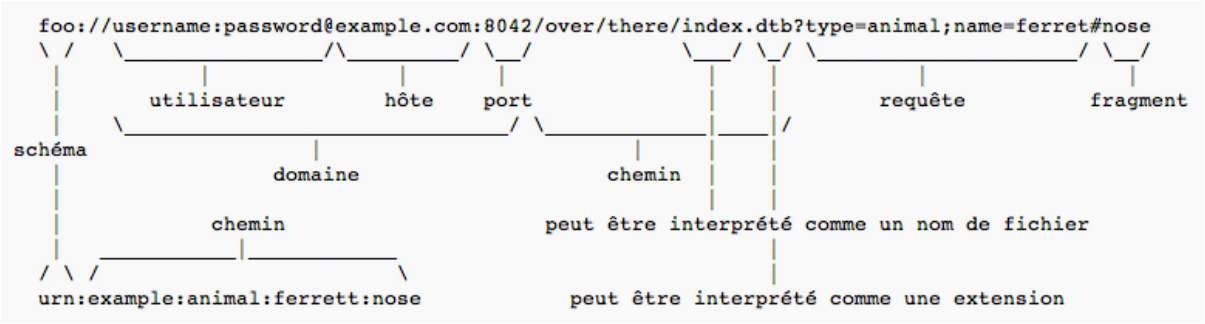


Figure 0-1 : Composition d'un URI [Wikipedia, 2012b]

URL (Uniform Resource Locator) : substitution du terme *adresse web,* désigne une chaine de caractères utilisée pour adresser les ressources du web.

User-agent : application cliente permettant la consultation de sites web. Il s'agit souvent de navigateurs, lecteurs d'écran,…

Wildcard : type de caractère informatique utilisé lors de la recherche d'un mot ou d'une expression incomplète.

XRI (eXtensible Resource Identifier) : schéma et protocole de résolution fournissant un format universel pour des identifiants abstraits de façon à ce qu'ils puissent être partagés à travers un grand nombre de domaines, répertoire et protocoles d'interactions.

1 Introduction

Depuis l'apparition du Web 2.0, la toile offre un panel de fonctionnalités de plus en plus large pour une expérience utilisateur toujours plus perfectionnée. Il est, en effet, de plus en plus commode de trouver, sur certains sites web, un bouton intitulé « Se connecter avec Facebook ». Ce bouton vous permet, comme son nom l'indique, de vous connecter avec vos identifiants Facebook sur les sites disposant de cette fonctionnalité.

L'objectif recherché est, parmi d'autres, l'authentification unique : se connecter une seule fois avec un fournisseur d'identité et avoir ainsi accès à une multitude de sites sécurisés.

Le principal problème au niveau du web est de savoir avec qui on est en train d'interagir réellement. Malgré certaines initiatives, il existe toujours un besoin d'un système vérifiant l'identité. Au vu du nombre important de systèmes effectuant l'authentification unique, nous avons réduit le champ de la recherche aux systèmes à communauté large ayant une approche fédérative ou la permettant et étant destinés à un environnement web.

Hormis l'aspect authentification unique, nous aborderons également les autorisations et leurs délégations, les différentes fonctionnalités qu'offrent certains de ces modules et l'accès aux ressources protégées. Ces aspects vont souvent de pair avec l'authentification unique.

Le travail effectué dans le cadre de ce mémoire s'est déroulé en plusieurs étapes. La première, la plus importante, consiste en une étude détaillée des protocoles. De cette étude, un effort de synthèse a été réalisé, où les éléments caractéristiques des différentes spécifications ont été mis en avant et illustrés dans ce texte. Ce mémoire présente un condensé de ces systèmes complexes, en vue d'en retirer l'information appropriée, afin d'en donner les avantages, les inconvénients, les points forts et les points faibles, ainsi que d'apporter une vue plus approfondie sur les aspects sécurité, vie privée et cas d'utilisation. Enfin, ce mémoire a été l'occasion de faire une étude comparative des différents protocoles et de proposer une grille de comparaison, ainsi qu'une méthode de sélection d'un système sur base de celle-ci.

La suite de ce travail est organisée comme suit.

Le **chapitre 2** retrace l'état de l'art. Il rappellera les différents aspects de l'identité, y compris les définitions des concepts utilisés. Il replacera le champ couvert par ce mémoire dans son contexte.

Nous poursuivrons, au **chapitre 3**, par une étude du protocole OAuth dans sa version 1.0a et 2.0.

L'implémentation de ce protocole, dans sa version 2, sera ensuite analysée au sein de la plate-forme Facebook Connect au **chapitre 4**.

Le **chapitre 5** abordera une nouvelle spécification, OpenID 2.0, avec une approche quelque peu différente et de nouvelles fonctionnalités intéressantes.

Au cours du **chapitre 6** nous aborderons OpenID Connect 1.0. Cette spécification tente de regrouper les avantages des deux spécifications vues au chapitre 3 et 5.

Le standard SAML 2.0, incontournable dans ce domaine, vous sera également présenté au **chapitre 7**.

Enfin, une analyse comparative des différents systèmes étudiés sera présentée au **chapitre 8**. Elle reprend les différents concepts étudiés tout au long de ce mémoire dans différentes grilles comparatives et vous donne les outils nécessaires pour pouvoir choisir le système dont vous avez besoin.

Le **chapitre 9** conclura ce mémoire. Il apportera une réponse à la question reprise dans l'intitulé du mémoire.

2 Etat de l'art

2.1 Préambule

Avant d'entrer dans le vif du sujet, rappelons les quelques concepts généraux suivants qui seront utilisés tout au long de ce travail.

Une entité est une personne, une organisation ou tout objet demandant accès à une ressource numérique.

Un identifiant est une information permettant de référencer une entité de manière unique.

Un attribut est une donnée décrivant une caractéristique d'un sujet.

Un jeton est un identifiant utilisé pour accéder à des ressources protégées.

L'identité numérique est un ensemble cohérent et homogène de caractéristiques d'une entité permettant de la reconnaître, de manière unique, dans le domaine de l'informatique. Elle contient également l'information sur les relations d'une entité avec d'autres. Elle peut être interprétée comme la codification des noms de l'identité et des attributs d'une instance physique de sorte à faciliter le traitement des données. L'identité numérique nécessite naturellement des identifiants numériques : des chaines de caractères ou jetons uniques avec un champ d'application donné.

2.2 Authentification vs identification

L'authentification et l'identification sont deux concepts complètement différents bien qu'ils soient liés de manière étroite.

L'identification est l'action permettant de connaître l'identité d'une entité sans nécessairement la contrôler.

L'authentification est la vérification d'informations relatives à une personne ou à un processus informatique. L'authentification consiste donc à contrôler l'identité et à définir un moyen de la valider.

L'authentification peut se faire sur base de [Villacres] :

- « Ce que je sais » : information confidentielle telle qu'un mot de passe ;
- « Ce que je sais faire » : signature sur écran tactile ;
- « Ce que je suis » : empreinte digitale, empreinte rétinienne ;
- « Ce que je possède » : une carte d'identité, carte bancaire.

Seuls l'Etat, un organisme sous son autorité, ou une société commerciale offrant un service réel sont capables d'offrir simultanément l'authentification et l'identification d'un individu réel. En effet, ceux-ci exigent une pièce d'identité afin de confirmer que vous êtes, en effet, la personne que vous prétendez être. Ils sont donc les seuls capables à vous identifier en tant que telle. Alors que dans le cadre de l'inscription à un service sur internet, cette identité n'est que très rarement vérifiée.

2.3 Autorisation et délégation de l'autorisation

L'autorisation est le mécanisme par lequel un système détermine quel niveau d'accès un utilisateur authentifié possède effectivement, et ce, afin de sécuriser les ressources protégées par le système.

L'autorisation peut donc être résumée en :

- « Ce que je peux faire » : le niveau d'autorisation reçu ;
- « Ce à quoi j'ai accès » : la délégation des autorisations.

L'autorisation peut être déléguée, c'est d'ailleurs un des points forts de certains mécanismes d'authentification unique. La délégation d'autorisation donne le droit à un fournisseur de services d'accéder aux ressources d'un utilisateur.

2.4 Gestion de l'identité

La gestion de l'identité est un effort effectué afin de rendre les identités disponibles pour les humains, services et systèmes d'une façon sécurisée et respectueuse de la vie privée.

Un service de gestion de l'identité, au sens large, a pour rôle :

- d'attribuer un identificateur à chaque entité ;
- d'authentifier une entité ;
- de servir des attributs sur cette entité ;
- de donner accès à des données externes ;
- de donner accès à des autorisations.

Les défis de la gestion de l'identité sont les suivants :

- trouver un bon compromis entre sécurité, vie privée et facilité d'utilisation ;
- rendre les identités disponibles uniquement aux individus ou services concernés au bon moment et au bon endroit ;
- établir la confiance entre les tiers impliqués dans les transactions d'identité ;
- éviter les abus d'utilisation d'identité ;
- rendre l'approvisionnement des identités possible d'une façon mesurable, opérationnelle et avec un bon rapport qualité-prix.

Les sept lois de l'identité [Cameron, 2009] :

1. Contrôle et consentement de l'utilisateur : les systèmes d'identité ne doivent révéler que l'information de l'identification pour laquelle ils ont reçu le consentement de l'utilisateur ;
2. Divulgation minimale : la solution divulguant le moins d'information identifiante et limitant au mieux son utilisation est une solution stable à long terme ;
3. Divulgation uniquement aux entités qui en ont besoin : les systèmes d'identification numérique doivent être conçus de façon à ne divulguer les informations qu'aux tiers ayant un rôle nécessaire et justifié pour une identité donnée ;
4. Identité ciblée : un système d'identification universel doit aussi bien supporter des identifiants « omnidirectionnels » pour leur utilisation par des entités publiques que des identifiants « unidirectionnels » pour l'utilisation par des entités privées ;
5. Pluralité des technologies des systèmes d'identité: un système d'identification universel doit canaliser et se servir de la collaboration de plusieurs technologies d'identification gérées par des fournisseurs d'identité multiples ;
6. Prise en compte du facteur humain : le système d'identification universel doit définir comment un utilisateur peut devenir un composant d'un système distribué en intégrant des mécanismes de communication homme-machine non ambigus. Il offre ainsi une protection contre les attaques d'usurpation d'identité ;
7. Expérience similaire quel que soit le contexte : unifier les différentes identités pour permettre à l'utilisateur une expérience consistante et simplifiée.

Différents acteurs prennent part au mécanisme de la gestion de l'identité :

- **le fournisseur d'identité**: acteur fournissant les identités au sujet. Ses tâches sont les suivantes :
    - générer et assigner les attributs d'identité spécifiques à un sujet ;
    - lier un attribut d'identité d'un sujet à d'autres attributs de celui-ci ;
    - générer des assertions concernant les attributs d'un sujet ;

- fournir des informations d'identification contenant des attributs et des assertions.

Synonymes : IdP (Identity Provider), OP (OpenID Provider), serveur d'autorisation, serveur (dans les communications de type client-serveur).

- **fournisseur de services :** acteur exigeant la soumission d'identifiants valides de la part de l'utilisateur afin de fournir le service désiré. Les rôles du fournisseur de services sont les suivants :
  - savoir déterminer dans quelle mesure il peut faire confiance aux identifiants de l'utilisateur et aux assertions reçues ;
  - être capable de vérifier l'identité et les attributs avec les fournisseurs d'identité ;

  Synonymes : RP (Relying Party), SP (Service Provider), consommateur, service, client (dans les communications de type client-serveur),…

- **contrôleur :** service répressif et/ou organisme(s) de réglementation qui peuvent avoir besoin d'un accès à l'information d'identité (via des journaux d'évènements,…) à des fins d'investigations. Bien que ce rôle existe, cette notion ne sera pas reprise dans le cadre de ce mémoire.

- **utilisateur/sujet :** personne souhaitant accéder aux services fournis par le fournisseur de services. A cette fin, elle utilisera un user-agent. Il s'agit bien souvent d'un navigateur. Le lien entre l'utilisateur et le navigateur est étroit étant donné que l'utilisateur a besoin de celui-ci pour communiquer avec les autres acteurs. Cette distinction est faite uniquement dans certaines spécifications, quand cela est nécessaire. Synonymes : client, user-agent, utilisateur, propriétaire de la ressource, navigateur, sujet,…

Le client est présent dans deux catégories d'acteurs : le fournisseur de services et l'utilisateur. En effet, dans certains cas de figure, le fournisseur de services est assimilé à l'utilisateur ou vice-versa. Un éclaircissement permettant de lever l'ambiguïté sera fourni dans l'introduction du chapitre si nécessaire.

La Figure 2-1 illustre les relations entre les différents acteurs.

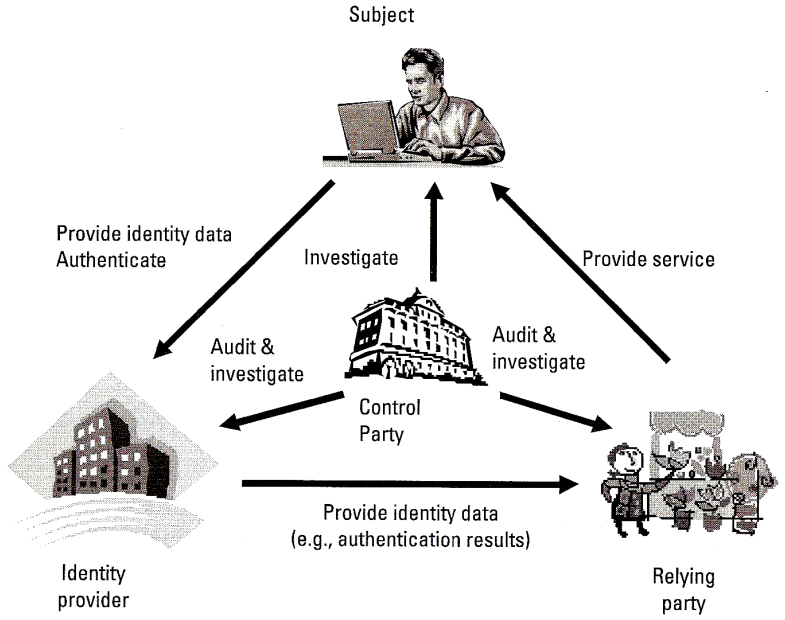


Figure 2-1 : Acteurs de la gestion de l'identité et leurs relations [Bertino et Takahashi, 2011]

Plusieurs rôles peuvent être couverts par la même entité (Facebook peut être à la fois un fournisseur d'identité et un fournisseur de services). Il est également possible que plusieurs fournisseurs d'identité établissent un lien de confiance entre eux avec le consentement de l'utilisateur afin de s'échanger les informations dont ils disposent.

2.5 Authentification unique

L'authentification unique, souvent appelé « Single Sign On » dans la littérature, est une méthode permettant à un utilisateur de s'authentifier une seule fois pour avoir accès à une multitude de sites web sécurisés.

Il existe 3 domaines dans lesquels l'authentification unique s'effectue:
- l'authentification unique en entreprise (ESSO) ;
- l'authentification unique multi-domaines ;
- l'authentification unique accessible par le web.

Le sujet de ce mémoire abordera principalement le dernier domaine cité.

Les objectifs de l'authentification unique sont les suivants:
- simplifier la gestion des mots de passe : plus l'utilisateur doit gérer de mots de passe, plus il aura tendance à utiliser des mots de passe similaires ou simples à mémoriser, abaissant par la même occasion le niveau de sécurité que ces mots de passe offrent.
- simplifier et centraliser la gestion des données personnelles : les modifications ne doivent plus s'effectuer qu'à un seul endroit ;
- permettre à l'utilisateur un certain contrôle sur la confidentialité de ses données ;
- simplifier la gestion et le développement des sites web : se concentrer sur le business, l'authentification étant gérée ailleurs ;
- simplifier la définition et la mise en œuvre de politiques de sécurité.

Comme le Professeur J-N Colin le stipule dans son cours [Colin, 2010] :
« Pour arriver à atteindre son objectif, l'authentification unique a besoin de :
- *standards flexibles et extensibles pour la représentation des données ;*
- *protocoles d'échanges d'informations ;*
    - *indépendants de la technologie sous-jacente ;*
    - *préservant la confidentialité des données ;*
    - *interopérables ;*
- *mécanismes de gestion de la confiance. »*.

L'authentification unique, aussi intéressante qu'elle soit, reste très dangereuse. En cas d'attaque réussie d'un tiers malintentionné, cette personne aurait accès à l'intégralité des services auxquels l'utilisateur avait accès. Les pertes peuvent donc être importantes ! La mise en œuvre de l'authentification unique requiert donc une attention particulière et primordiale concernant la sécurité et les méthodes d'authentification utilisées.

Les trois approches de mise en œuvre d'un système d'authentification unique sont les suivantes :
- **approche centralisée** : base de données globale et centralisée contenant tous les utilisateurs et destinée à des services dépendant d'une même entité. Cela permet également de centraliser la gestion de la politique de sécurité. Exemple : au sein d'une société ou d'une entreprise ;
- **approche fédérative** : chaque service dispose d'une partie de l'information concernant un utilisateur mais la partage avec d'autres services partenaires.

Cette approche a été développée pour répondre à un besoin de gestion décentralisée des utilisateurs, où chaque service partenaire désire conserver la maîtrise de sa propre politique de sécurité. L'approche fédérative améliore donc la portabilité des informations d'identité à travers des domaines autonomes en permettant aux utilisateurs d'un domaine d'accéder aux données ou au système d'un autre domaine en toute sécurité et de façon transparente. Le tout sans la nécessité d'administrer l'utilisateur de façon redondante. Exemple : sites marchands indépendants d'un point de vue commercial et organisationnel,…

- **approche coopérative** : chaque utilisateur dépend d'une des entités partenaires qui l'authentifie. Comme dans l'approche fédérative, chaque service du réseau gère indépendamment sa propre politique de sécurité. Exemple : dans les administrations, les universités,…

La notion de **fédération d'identités** :

« La fédération tente d'améliorer la convivialité pour les utilisateurs en limitant le nombre de fournisseurs d'identité. Plus le nombre de systèmes auxquels les utilisateurs peuvent accéder sans s'authentifier de nouveau est élevé, plus la convivialité est grande pour eux.

La fédération agit comme un mécanisme périphérique qui se situe en bordure du réseau et partage les informations d'identité avec d'autres mécanismes de fédération avec lesquels il existe une relation de confiance. La technologie de fédération s'appuie sur la confiance accordée aux pratiques d'authentification, d'administration et de confirmation d'identité d'une entité membre, pour certifier que les utilisateurs ou services d'applications peuvent accéder à une ressource externe, et vice versa » [EGILIA, 2010].

Le présent mémoire portera principalement sur l'authentification unique avec une approche fédérative dans un environnement web. Il s'agit principalement de modules d'authentification basés sur des standards ou « à communauté large » offrant également un service d'autorisation.

OAth n'étant pas très répandu, il ne sera pas étudié dans ce mémoire aux dépens d'autres systèmes d'authentification unique.

2.6 Déconnexion unique

Le mécanisme de déconnexion unique, souvent appelé « Single Sign Out » dans la littérature, est une méthode permettant à un utilisateur de se déconnecter en un seul clic de tous les sites auxquels il était connecté avec le même identifiant. Ce mécanisme n'est pas toujours abordé par les protocoles d'authentification unique. Il doit idéalement l'être dans l'implémentation de tout protocole.

2.7 Types de document

Afin de pouvoir clairement faire la distinction entre les différents types de documents existants, voici une courte description de ceux-ci:

- **norme** : spécification technique émise par un organisme de normalisation et destinée à harmoniser l'activité d'un secteur industriel ;
- **protocole** : spécification de plusieurs règles/ensemble de normes décrivant le comportement d'une technologie (un type de communication) particulière ;
- **standard** : étape de l'avancement d'une norme, norme de fait répandue par son utilisation générale. Lorsqu'une norme est acceptée par l'organisme de normalisation, elle devient un standard. Un standard n'est pas forcément détaillé dans un document officiel ;
- **spécification** : document technique reprenant un ensemble explicite d'exigences à satisfaire pour un produit ou service.

2.8 Types d'attaque

Cette section a pour objectif d'établir une liste des différentes menaces que l'on pourrait être amené à rencontrer dans le cadre de l'authentification unique. Les recommandations liées à ces menaces, si elles ne sont pas générales, sont spécifiques aux protocoles et sont donc reprises au sein de chacun des chapitres. Cette section offre donc un aperçu et une brève explication de l'objectif recherché pour chacune des attaques ainsi que ce en quoi elle consiste.

Rappelons également que selon un rapport de l'institut SANS [SANS, 2007], les principales failles de sécurité présentes du coté client proviennent essentiellement des navigateurs web alors que coté serveur il s'agit des applications web. Le domaine étudié dans ce travail est donc plus propice aux failles de sécurité.

Les attaques sont classées par ordre alphabétique.

Attaque par cryptanalyse

Cette attaque consiste à tenter de déchiffrer un message ayant été chiffré avec une clé. Une analyse de la clé est faite sur base de certains critères suivant le type d'analyse et le type d'algorithme utilisé pour le chiffrement de la clé.

Un exemple de ce type d'attaque est l'**attaque temporelle** permettant à la personne malintentionnée d'obtenir certaines informations en chronométrant la différence de temps écoulée entre un déchiffrement avec succès et la vérification incorrecte d'un message signé. Cette information peut être utilisée pour réduire la longueur de la clé ou la longueur du chiffrement utilisé.

Attaque CSRF (Cross-Site Request Forgery)

Pour rappel les attaques de ce type consistent à rediriger l'utilisateur vers un site personnel forçant l'utilisateur à effectuer une action potentiellement dangereuse à son insu. Ce type d'attaque se sert donc de l'utilisateur comme déclencheur afin qu'il devienne complice sans en être conscient. Cette technique permet de contourner un grand nombre de systèmes d'authentification unique.

Attaque par fixation de session

Les attaques par fixation de session exploitent la vulnérabilité d'un système qui permet à quelqu'un de déterminer l'identifiant de session d'une autre personne.

Attaque par force brute

Type d'attaque utilisée pour trouver le mot de passe d'un utilisateur. Le concept est assez simple puisqu'il consiste à soumettre de manière exhaustive un ensemble de combinaisons de mot de passe jusqu'à trouver la combinaison correcte.

Au plus la complexité du mot de passe est élevée et longue, au plus le temps nécessaire pour trouver la bonne combinaison sera grand.

Attaque par rejeu

Forme d'attaque réseau dans laquelle une transmission est répétée par une tierce partie ayant préalablement intercepté les paquets. On peut imaginer capturer des informations de sécurité tel que des jetons d'accès, des assertions,… et ensuite les réutiliser en les rejouant de la même façon afin de tenter d'accéder à certaines ressources.

Attaque de l'homme du milieu

Aussi appelé attaque « Man in the Middle » (MitM), il s'agit d'une attaque ayant pour but d'intercepter les communications entre deux parties, sans que ni l'une ni l 'autre ne s'en rendent compte. Ce type d'attaque est possible lorsque les données d'identité sont exposées de manière involontaire. On pourrait imaginer un pirate informatique

s'ajoutant des accès non autorisés en modifiant le champ d'application concerné pour l'accès à des ressources protégées.

Clickjacking

Technique malveillante visant à pousser un internaute à fournir des informations confidentielles ou à prendre le contrôle de son ordinateur en le poussant à cliquer sur des pages apparemment sûres.

Déni de service

Cette attaque vise principalement la disponibilité du service. Le déni de service, appelé « Denial of services » (DoS) en anglais, consiste à rendre indisponible l'accès aux informations et services en saturant le système par un grand nombre de requêtes concurrentes.

Ecoute du réseau

Il y a plusieurs types d'attaque d'écoute envers les canaux de communication représentant des menaces immédiates pour la confidentialité des données. Les messages du protocole HTTP peuvent facilement être capturés par un public à qui ils n'étaient pas destinés. Cette méthode offre donc une vue sur les données d'identité de la personne concernée.

Hameçonnage

Technique utilisée par des fraudeurs pour obtenir des renseignements personnels dans le but de perpétrer une usurpation d'identité. Souvent elle consiste à manipuler un URI de redirection afin de rediriger l'utilisateur vers un autre URI. L'attaque dans le cadre de l'authentification unique consiste à rediriger l'utilisateur vers un tiers en se faisant passer pour le fournisseur d'identité de l'utilisateur.

Usurpation d'identité

Cette attaque a pour but de légitimer la communication entre des nœuds qui ont été remplacés par d'autres. Dans ce type d'attaque la principale faiblesse se situe au niveau de l'utilisateur qui représente sans doute le nœud le plus faible de la chaine. L'objectif de cette attaque est d'obtenir un maximum d'informations d'identité pouvant entrainer de sérieuses pertes financières et sociales.

Le « déguisement » d'un serveur, l'hameçonnage par un tiers malveillant font donc également partie des attaques d'usurpation d'identité. En effet, un serveur malintentionné peut se faire passer pour un serveur légitime par différents moyens.

3 OAuth

3.1 Introduction

OAuth est un protocole libre issu de la communauté OpenID, qui fut créé par un groupe de travail composé de développeurs web qui recherchaient une solution à la délégation d'accès à des ressources protégées. Certains services web, le cloud-computing et d'autres services partagés, font souvent appel à ce type de protocole. La première version stable [Hammer, 2010] de OAuth est apparue en octobre 2007, elle a été suivie d'une révision en juin 2009 pour aboutir à une version 1.0a et depuis, une version 2.0 en version brouillon, décrite ci-après, est toujours en cours de finalisation. Ce protocole a déjà été implémenté par de nombreux grands acteurs tels que Twitter, Yahoo, Facebook,…

OAuth est basé sur le modèle d'authentification client-serveur auquel on a ajouté un nouvel acteur qui n'est autre que le propriétaire de la ressource que le client souhaite atteindre. Le principe est simple : le client souhaite obtenir les accès à une ressource détenue par son propriétaire sur le serveur. Pour ce faire, le client demande une autorisation au propriétaire. OAuth effectue également des contrôles d'identité sur le client effectuant la requête.

Le concept d'OAuth consiste à introduire une couche d'autorisation, séparant ainsi le rôle du client de celui du propriétaire de la ressource. Cette couche fournira des identifiants différents de ceux du propriétaire de la ressource.

Voici à titre d'exemples quelques cas d'utilisation d'OAuth : applications incluant un paiement, génération d'assertions, gestion de contenu, échange de jetons d'accès, messages,… L'ensemble de ces cas d'utilisation est détaillé dans un document de l'IETF [Fletcher et al, 2011].

3.2 OAuth v1.0a

3.2.1 Type et composition

Les trois acteurs principaux impliqués, dans la spécification OAuth v1.0a, sont :

- **le client** : partie qui souhaite accéder à la ressource. Il ne s'agit donc pas ici de l'utilisateur mais bien de la partie cliente de la communication client-serveur. Il s'agit donc du **fournisseur de services** ;
- **le serveur** : partie qui héberge la ressource. Correspond au **fournisseur d'identité** dans cette spécification (une distinction entre le fournisseur d'identité et le serveur hébergeant la ressource est faite dans OAuth v2) ;
- **le propriétaire de la ressource** : représente l'**utilisateur** donnant accès à ses ressources privées.

Afin de faciliter la compréhension du protocole et éviter toute confusion, le propriétaire de la ressource est appelé utilisateur tout au long de ce chapitre.

OAuth a pour rôle de fournir:

- une méthode aux clients pour accéder aux ressources d'un serveur au même titre que le propriétaire de la ressource (délégation des autorisations d'accès);
- un processus d'autorisation, via un mécanisme de redirection, permettant à un utilisateur de donner accès à la ressource présente sur le serveur sans fournir ses identifiants au tiers.

Le principe de OAuth se base sur l'accès à des ressources protégées à l'aide d'un jeton. Un jeton, est un identifiant unique émis par le serveur et utilisé par le client pour permettre au propriétaire de la ressource d'authentifier les requêtes vers la ressource pour laquelle l'autorisation a été obtenue. Il est fourni par le propriétaire de la ressource et consommé par le client. Ce jeton représente le secret partagé entre les deux acteurs.

Les avantages de l'utilisation d'un jeton :
- le propriétaire de la ressource ne doit pas partager ses identifiants ;
- possibilité de définir le champ d'application du jeton ;
- possibilité de définir une durée de vie au jeton et donc un temps d'accès limité aux ressources ;
- possibilité de révoquer le jeton de façon indépendante.

L'inconvénient majeur du jeton est lié à la sécurité. Si le jeton n'est pas bien sécurisé, un pirate peut se l'approprier et accéder ainsi aux informations du propriétaire.

3.2.2 Concepts de base

OAuth fournit donc :
- un processus d'accès du client aux ressources de l'utilisateur (délégation de l'autorisation) par l'intermédiaire de redirection de l'agent client. Cela implique :
  - une authentification directe avec le serveur possédant les ressources ;
  - un approvisionnement de jetons aux clients pour l'utilisation de la méthode d'authentification ;
- une méthode pour effectuer des requêtes authentifiées HTTP en utilisant deux ensembles d'identifiants:
  - identifiants du client effectuant la requête ;
  - identifiants du propriétaire de la ressource par l'intermédiaire duquel la requête va être exécutée.

3.2.3 Etude du protocole

3.2.3.1 Remarques préliminaires

Avant de présenter le fonctionnement interne du protocole, il nous faut définir la notion d'identifiants. Les identifiants, appelé « credentials » dans la littérature, représentent un couple constitué d'un identifiant unique et d'un secret partagé. OAuth définit trois classes d'identifiants :

1. **identifiants du client** : identification/authentification du client. Composés de la clé du client et du secret partagé ;
2. **identifiants temporaires** : obtenus en effectuant une requête d'autorisation:
   - limités dans le temps et révoqués après utilisation ;
   - possibilité pour le propriétaire de la ressource de révoquer les identifiants du jeton après qu'ils aient été émis aux clients ;

   Composés d'un jeton temporaire et du secret partagé.
3. **identifiants du jeton :** l'autorisation d'accès. Composés du jeton d'accès et du secret partagé.

Le serveur doit posséder trois URL différentes pour fournir ces différents identifiants. Les 3 étapes reprises en vert sur la Figure 3-1 représentent les différentes phases pendant lesquelles ces identifiants sont obtenus (dans l'ordre mentionné ci-dessus).

3.2.3.2 *Le flux d'authentification*

Les échanges effectués au sein du processus sont numérotés en rouge sur le schéma de la Figure 3-1 et expliqués en dessous de celui-ci.

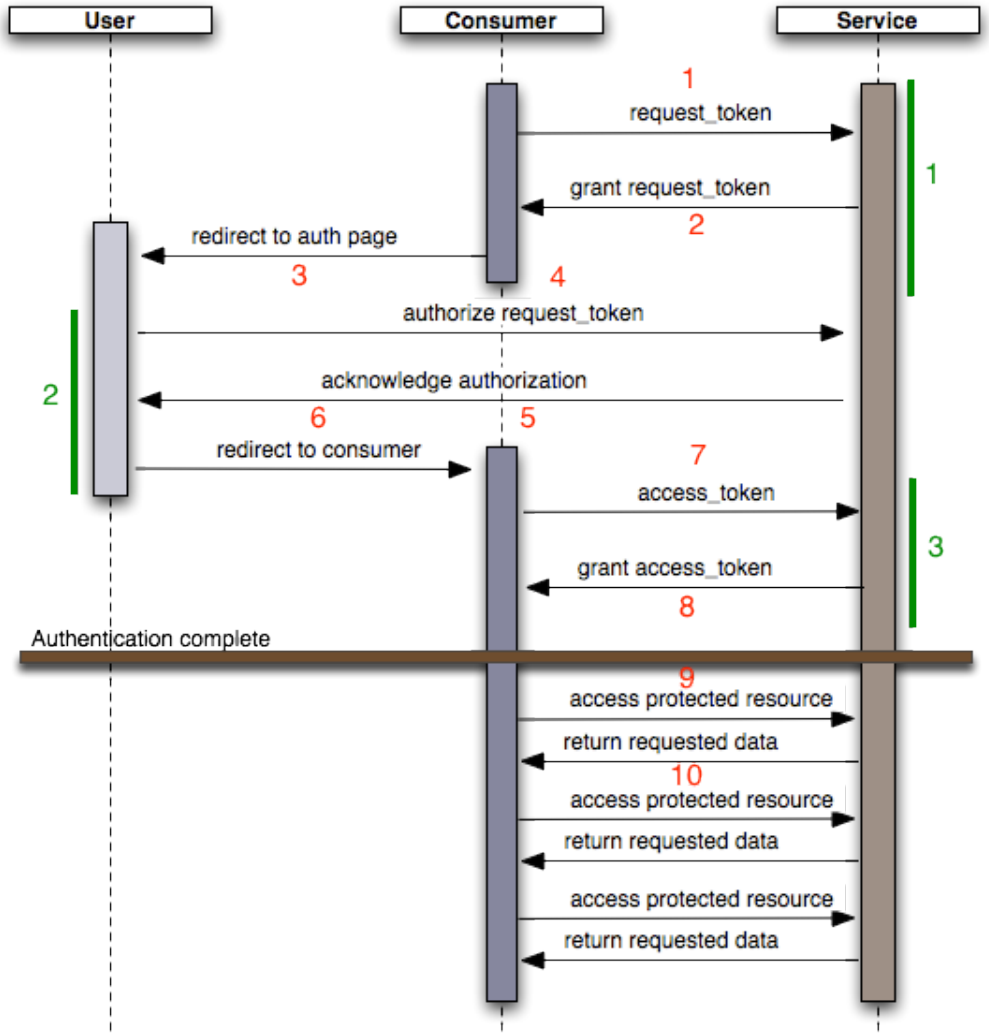


Figure 3-1 : Flux d'authentification OAuth v1.0a [Trenka, 2009]

1. L'utilisateur ne souhaitant pas partager ses identifiants, le client (repris sous le terme « Consumer » dans la Figure 3-1) initialisera le processus en établissant une connexion avec le serveur hébergeant la ressource afin d'obtenir ses identifiants temporaires en utilisant la méthode de signature requise et définie par celui-ci. Lorsque le client effectue la requête, il utilisera ses propres identifiants (identifiants client). Cette communication sera effectuée avec une connexion sécurisée (SSL, TLS,…).
2. Le serveur, après vérification de la requête, répondra en fournissant des identifiants temporaires.
3. Le client doit, ensuite, autoriser la requête à destination de l'URI d'autorisation du serveur à l'aide des identifiants temporaires reçus à l'étape précédente. Le client redirige alors le user-agent vers le serveur, afin d'obtenir les autorisations nécessaires de la part de l'utilisateur pour pouvoir fournir le service demandé.
4. L'utilisateur s'authentifie sur le serveur, si ce n'est déjà fait, et autorise le client à accéder à la ressource demandée.
5. Le serveur contrôle l'identité de l'utilisateur et va ensuite le rediriger vers un URI de redirection défini précédemment par le client ou en utilisant un tout autre mécanisme si cet URI n'était pas initialisé.

Lorsque l'on demande à l'utilisateur d'autoriser la requête d'accès, le serveur devrait lui afficher l'identité du client ayant demandé l'accès. Il doit également mentionner si les différentes informations ont été vérifiées afin de s'assurer que l'utilisateur est bien celui-ci qu'il prétend être. Le serveur doit finalement générer un code de vérification qui sera fourni au client.
Si le client n'est pas en mesure de fournir un URI pour la réponse à la requête, le serveur doit afficher la valeur du code de vérification et demander à l'utilisateur d'informer le client que le processus d'autorisation est achevé.

6. Redirection effective du user-agent vers le client à l'aide d'un URI défini durant la première étape.
7. Le client effectue maintenant une requête sur l'URL prévue à cet effet afin d'obtenir les identifiants du jeton. A cette fin, il doit s'authentifier en utilisant d'une part sa propre identité et d'autre part le code de vérification reçu à l'étape précédente.
8. Le serveur doit, à cette étape :
    - exiger une connexion sécurisée ;
    - vérifier la validité de la requête (ceci est fait en vérifiant que le propriétaire de la ressource a effectivement autorisé l'approvisionnement des identifiants temporaires, que ceux-ci sont valides, non expirés et qu'ils n'ont pas été utilisés précédemment) ;
    - valider le code de vérification fourni par le client.

   Si les éléments précédents sont fournis et valides, les identifiants du jeton pourront alors être délivrés dans la réponse. A cette étape le client a reçu tous les éléments nécessaires pour accéder à la ressource partagée. L'autorisation de l'utilisateur est à présent représentée sous la forme d'un jeton d'accès et de son secret correspondant. L'autorisation a donc été déléguée.
   Le serveur fournit les éléments demandés et révoque les identifiants préalablement créés.
9. Le client est maintenant prêt à accéder aux ressources de l'utilisateur hébergées sur le serveur par l'intermédiaire de son jeton d'accès et de son secret. Cette requête est appelée « requête authentifiée ». Une requête authentifiée possède plusieurs paramètres relatifs au protocole. Le client assigne les valeurs appropriées à ces paramètres en utilisant ses identifiants de client, le jeton d'accès, l'horodatage actuel et un mot unique créé pour l'occasion (appelé *nonce*).
10. Le serveur reçoit les requêtes authentifiées et les valide :
    - en recalculant la signature de la requête et en la comparant à la valeur reçue du client ;
    - en cas d'utilisation de méthodes de signature, en s'assurant que la combinaison des identifiant par jeton, de l'horodatage actuel, et du mot unique créé à l'occasion (*nonce*) n'a pas déjà été utilisée auparavant ;
    - si un jeton est présent, en vérifiant le champ d'application de ce dernier et le statut de l'autorisation cliente ;
    - en vérifiant que la version d'OAuth, si elle est mentionnée, correspond bien à la version 1.0.

    Si ces conditions sont remplies, il renverra les attributs demandés.
    Les requêtes authentifiées possèdent deux types d'ensembles d'identifiants. Afin que le client prouve qu'il est bien le propriétaire des identifiants qu'il possède, le serveur comparera le secret partagé dans les deux ensembles. OAuth n'exige pas une méthode de signature particulière et permet une implémentation libre du serveur quant à la méthode utilisée.

3.2.4 Considérations en termes de sécurité

La majorité des menaces est liée, non pas au protocole lui-même, mais aux polices et procédures qui l'entourent. Voici les principaux risques et points sur lesquels il faut apporter une attention particulière :

- la sécurité des requêtes authentifiées avec les signatures « RSA-SHA1 » repose uniquement sur le secret de la clé privée du client. Cette méthode de signature présente certaines faiblesses cryptographiques [Schneirer, 2005] ;
- aucune garantie sur la confidentialité des requêtes (il y a bien une garantie d'intégrité) → SSL ou TLS conseillé ;
- l'authenticité du serveur n'étant à aucun moment vérifiée, une partie adverse peut profiter de cette faille pour intercepter les requêtes du client et y répondre de façon trompeuse ou incorrecte → SSL ou TLS conseillé ;
- l' « authorization scheme », paramètre facultatif, dépend des en-têtes « Authorization » et « WWW-Authenticate » pour distinguer le contenu authentifié afin de le protéger. Le proxy et la mise en cache peuvent ne pas fournir la protection adéquate pour ces en-têtes lorsque ces derniers sont ignorés ;
- le serveur doit pouvoir accéder aux secrets partagés du client et du jeton sous la forme d'un texte brut. Il est donc critique, pour le serveur, d'offrir une protection élevée pour ses secrets ;
- la sécurité des identifiants de l'utilisateur est primordiale et dans des environnements peu protégés des personnes malintentionnées pourraient très facilement retrouver les identifiants de l'utilisateur. Pour cette raison, il est conseillé que le serveur n'utilise pas uniquement les identifiants du client pour vérifier son identité mais également une information personnelle (adresse IP, adresse MAC de l'utilisateur,…) ;
- les propriétaires des ressources peuvent devenir insensibles aux redirections vers des sites ou le couple utilisateur/mot de passe est exigé. Ceci profite évidemment aux personnes malintentionnées → Le serveur doit enseigner à l'utilisateur à vérifier son authenticité (celle du serveur) ;
- le champ des autorisations n'est pas couvert par ce protocole. Le serveur devra donc, quand cela est nécessaire, implémenter ce type de mécanisme et vérifier que les accès demandés, ainsi que les risques encourus, ont bien été compris par l'utilisateur ;
- les secrets partagés doivent être suffisamment complexes et aléatoires que pour ne pas être découverts par un type d'attaque par force brute durant leur durée de validité. Le générateur de nombre aléatoire devra être fiable et non sujet à faciliter ce type d'attaque ;
- le déni de service doit être pris au sérieux car il existe plusieurs possibilités d'arriver à amener le serveur à utiliser toutes les ressources dont il dispose: utilisation de plusieurs nonces rapidement épuisant les différentes possibilités, calcul pour vérifier la signature des requêtes entrantes,… ;
- toute requête qui ne serait pas couverte par le mécanisme de la signature de la chaîne doit faire l'objet d'une étude approfondie en termes de sécurité ;
- le serveur doit considérer les meilleures pratiques en termes de prévention contre les attaques CSRF sur tous les endpoints du protocole d'autorisation mais également concernant les URI de réponse à la requête ;
- se protéger contre les attaques de type clickjacking en utilisant entre autres un navigateur spécifique implémentant des techniques d'anti-framing (technique de programmation offrant la possibilité de diviser la fenêtre d'un navigateur web en plusieurs cadres autonomes), une invitation à réintroduire son mot de passe avant de fournir les jetons OAuth ;

- se protéger contre les processus d 'autorisations répétés automatiquement. En effet, si les identifiants du client sont compromis, ceci augmente les risques d'attaques. La personne malintentionnée pourrait rediriger l'utilisateur vers le serveur avec une requête d'autorisation sans avoir obtenu, au préalable, l'accord explicite de l'utilisateur. Il est possible de limiter cette pratique en minimisant le champ d'action du jeton.

3.2.5 Failles de sécurité

Les failles de sécurité sont souvent liées à la mauvaise mise en œuvre des spécifications, leur mauvaise interprétation, leur complexité mais aussi, parfois, parce que celles-ci ne sont pas toujours suffisamment exigeantes en matière de sécurité (l'exemple le plus évident étant la sécurité des couches de communication qui n'est pas exigée).

Il est donc important de tenir à l'œil toute implémentation de OAuth qui ne serait pas sécurisée par une couche de transport telle que TLS ou SSL. Mais il est également du devoir de l'utilisateur de s'assurer de l'identité et de l'authenticité du client (fournisseur de services) demandant l'accès à cette ressource.

L'autorisation fournie par l'utilisateur n'est donc pas un simple geste anodin bien que souvent considéré comme tel par bon nombre d'utilisateurs pourtant souvent avertis.

Le protocole OAuth, même si toutes les mesures de précaution ont été prises et correctement interprétées, n'est cependant pas infaillible… Twitter en a fait l'expérience et a été forcé, pour des raisons de sécurité, de désactiver OAuth. Le réseau social montrait du doigt des problèmes de sécurité avec le protocole OAuth lors d'attaques de fixation de session [Hammer, 2009]. Les problèmes de sécurité ont été exagérés et étaient principalement liés à l'implémentation même de OAuth par Twitter. Il a permis cependant de pointer tout de même une faille de sécurité du protocole. Pour reprendre l'explication de hueniverse.com [Hammer, 2009] et en résumé, même si une application :

- s'assure que le compte revenant suite à une réponse à une requête est bien celui de l'utilisateur et pas le compte de celui qui a initié la communication ;
- vérifie que l'utilisateur qui a démarré le processus est celui qui l'a terminé ;
- utilise une réponse à une requête immutable configurée à l'enregistrement ;

le protocole est toujours exposé à une attaque temporelle. Il n'y a, en effet, aucune possibilité de savoir qu'une personne ayant donné l'autorisation de l'accès à la ressource est la même personne qui revient la consulter.

3.2.6 Synthèse

En conclusion OAuth v1.0a est un protocole créé pour combler le manque de standard permettant de déléguer des accès et donc des autorisations par API. D'un point de vue sécurité, c'est est un protocole sûr, à condition qu'il soit implémenté en suivant à la lettre la spécification et en appliquant toutes les recommandations en termes de sécurité. OAuth 1.0a est, cependant, très limité au niveau des cas d'utilisation, puisqu'un seul flux y est décrit. Ce dernier point est une porte ouverte aux mauvaises implémentations si l'on s'écarte du flux de base.

Afin de pallier à d'autres problèmes de sécurité, la mise en place d'une couche de transport sécurisée (TLS ou SSL) est plus que vivement conseillée pour ne pas dire indispensable car ce protocole présente certaines faiblesses (discutées précédemment). Le développement doit être réfléchi et étudié au delà des aspects mentionnés dans la spécification. Le protocole est un peu trop laxiste en matière de sécurité et devrait être beaucoup plus strict.

3.3 OAuth v2

La section présente a été réalisée sur base du brouillon [Hammer et al, 2012] en version 2.22 soumis à l'IETF à titre de standardisation.

Les diagrammes de séquence présents dans ce chapitre et ceux, de même apparence, présents dans les suivants ont été réalisés à l'aide du site « Web Sequence Dia »[1].

OAuth v2 a pour principal objectif d'améliorer, par rapport à sa première version, trois de ses lacunes:

- la complexité des exigences cryptographiques de sa spécification pour l'authentification et les signatures ;
- la pauvre « expérience utilisateur » liée à l'unique flux (limité) d'obtention de jetons décrite dans la spécification. Il définira donc des options alternatives à la délivrance de jetons ;
- les performances à l'échelle : le protocole ne s'y adapte pas bien et pour cause il nécessite :
    - une gestion des états à différentes étapes ;
    - des identifiants temporaires peu utilisés ;
    - des identifiants de longue durée étant moins sécurisés et plus difficiles à gérer.

3.3.1 Type et composition

OAuth, dans sa seconde version, définit 4 acteurs principaux:

- le **propriétaire de la ressource** : entité capable d'autoriser l'accès à une ressource protégée. Comparable à l'**utilisateur** ;
- le **serveur** : entité hébergeant la ressource protégée, capable d'accepter et de répondre aux requêtes d'accès utilisant des jetons d'accès ;
- le **client** : application effectuant les requêtes d'accès aux ressources protégées à l'insu du propriétaire de la ressource et avec son autorisation. Comparable au **fournisseur de services** ;
- le **serveur d'autorisation** : le serveur fournissant les jetons d'accès au client après avoir authentifié avec succès le propriétaire de la ressource et obtenu son autorisation. Comparable au **fournisseur d'identité.**

OAuth en version 2, offre :

- plusieurs méthodes aux clients pour accéder aux ressources d'un serveur au même titre que l'utilisateur (délégation des autorisations d'accès) ;
- un processus d'autorisation, via un mécanisme de redirection, permettant à un utilisateur de donner accès à la ressource présente sur le serveur sans fournir ses identifiants au tiers.

Un serveur d'autorisation peut être identique au serveur hébergeant la ressource mais cette spécification envisage la possibilité qu'ils soient différents.
Un serveur d'autorisation est unique s'il peut fournir des jetons d'accès acceptés par plusieurs serveurs hébergeant la ressource.

3.3.2 Les nouveaux concepts

La version 2 d'OAuth inclut les nouveaux concepts suivants :

- **le serveur d'autorisation :** décrit précédemment ;

[1] L'outil est disponible ici : http://www.websequencediagrams.com/

- **la notion de jeton d'accès** a été quelque peu redéfinie afin d'y inclure les nouveaux concepts (différents types de flux, acteur supplémentaire,…) mais son rôle reste identique. Voici les principaux points d'attention :
    - le jeton est une chaîne de caractères insignifiante pour le client;
    - les jetons sont octroyés par le propriétaire de la ressource et renforcés par :
        - le serveur hébergeant la ressource ;
        - le serveur d'autorisation ;
    - le jeton peut indiquer un identifiant (présent à l'intérieur du jeton) utilisé pour retirer une information concernant l'autorisation ;
    - le jeton d'accès fournit une couche d'abstraction, remplaçant différentes constructions d'autorisations par un jeton unique compris par le serveur hébergeant la ressource. Cette abstraction permet de fournir des jetons d'accès plus restrictifs que l'autorisation utilisée pour les obtenir ;
    - les jetons d'accès peuvent avoir différents formats, structures, et méthodes d'utilisation sur base de la politique de sécurité du serveur hébergeant la ressource.

- **La notion de jeton d'actualisation :** les jetons d'actualisation sont fournis aux clients par le serveur d'autorisation. Ils sont utilisés pour obtenir un nouveau jeton d'accès quand le jeton courant est invalide ou expiré. Ils sont également utilisés pour obtenir des jetons d'accès supplémentaires avec un champ d'application proche ou identique. Ces jetons sont facultatifs et sont fournis en même temps que les jetons d'accès. Contrairement aux jetons d'accès, les jetons d'actualisation sont prévus pour être utilisés uniquement avec les serveurs d'autorisation et ne sont jamais envoyés aux serveurs hébergeant la ressource.

3.3.3 Concepts de base

Afin d'illustrer le concept de base de OAuth v2, le diagramme de séquence abstrait de la Figure 3-2, a été réalisé à partir des éléments présents dans sa spécification. Ce diagramme (et ceux qui suivent) ne contient pas les notions de communications synchrones/asynchrones puisqu'il ne représente que les interactions entre acteurs.

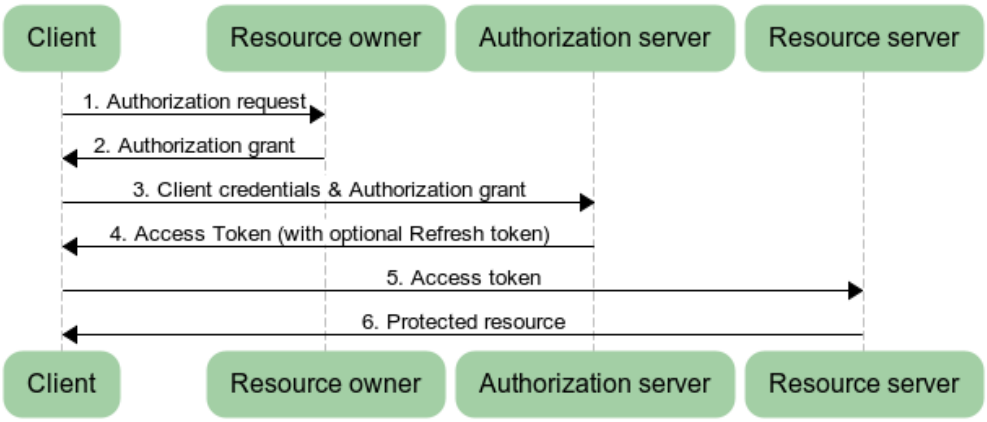


Figure 3-2 : Flux abstrait du protocole OAuth v2.0

1. Le client effectue directement une requête d'autorisation à destination du propriétaire de la ressource ;
2. Le client reçoit alors un identifiant représentant l'autorisation du propriétaire de la ressource. Cette autorisation est fournie en utilisant :
    - soit un des 4 types de flux d'autorisation parmi:
        - le flux « code d'autorisation » ;

        - le flux « octroi implicite » ;
        - le flux « identifiants par mot de passe du propriétaire de la ressource » ;
        - le flux « identifiants du client ».

      Le type de flux d'autorisation dépend de :
        - la méthode utilisée par le client pour faire la demande d'autorisation ;
        - les types de flux supportés par le serveur d'autorisation.
    - soit un mécanisme permettant de définir d'autres types.
3. Le client demande un jeton d'accès en s'authentifiant auprès du serveur d'autorisation et en présentant l'autorisation reçue préalablement du propriétaire de la ressource.
4. Le serveur d'autorisation authentifie le client, valide l'octroi des autorisations et fournit, ensuite, un jeton d'accès avec éventuellement un jeton d'actualisation.
5. Le client demande la ressource protégée au serveur qui l'héberge et s'authentifie auprès de lui en présentant le jeton d'accès.
6. Le serveur hébergeant la ressource valide le jeton d'accès et fournit, s'il est valide, les attributs demandés.

3.3.4 Etude du protocole

3.3.4.1 Enregistrement du client sur le serveur d'autorisation

Le serveur choisit le type de flux qu'il souhaite implémenter sur base de sa politique de sécurité. Le client devra donc :

- fournir des URI de redirection pour le client ;
- inclure l'ensemble des informations requises par le serveur d'autorisation ;
- spécifier son type parmi :
    - *confidentiel* : les clients sont capables de maintenir la confidentialité de leurs identifiants ou de sécuriser leur authentification. Les clients confidentiels se verront attribuer un ensemble d'identifiants utilisés pour l'authentification avec le serveur d'autorisation ;
    - *public* : les clients sont incapables de répondre aux exigences de type confidentiel. Le serveur d'autorisation peut mettre en place un moyen d'authentifier les clients de ce type. Par contre, il ne doit pas se fier à l'authentification publique cliente en termes d'identification du client.

La spécification a été créée autour des profils clients suivants :

- les applications web (les identifiants et les jetons d'accès sont hébergés sur le serveur web et ne sont pas accessibles par le propriétaire de la ressource) ;
- les applications basées sur le user-agent (les données du protocole et les identifiants sont accessibles et souvent visibles par le propriétaire de la ressource) ;
- les applications natives (les données du protocole et les identifiants sont accessibles par le propriétaire de la ressource).

3.3.4.2 Authentification du client

Les clients en possession d'un mot de passe peuvent utiliser l'authentification HTTP Basic [Franks et al, 1999] pour s'authentifier avec le serveur d'autorisation. L'identifiant du client est utilisé comme nom d'utilisateur et le mot de passe comme mot de passe. Le client peut également, s'il le souhaite et bien que cela ne soit pas recommandé, inclure ses identifiants dans le corps de la requête.

Le serveur d'autorisation doit exiger l'utilisation d'un mécanisme sécurisé de la couche transport lorsqu'il reçoit des identifiants en clair et doit se protéger de tout type d'attaque par force brute.

Le serveur d'autorisation peut supporter tout type de régimes d'authentification adaptés qui satisfait à ses exigences en termes de sécurité. Si tel est le cas, il devra définir une correspondance entre l'identifiant du client et le type d'authentification utilisé.

3.3.4.3 Les endpoints

Le processus d'autorisation, décrit à la suite de cette section, utilise deux endpoints :

- le **endpoint d'autorisation** : permet d'obtenir l'autorisation du propriétaire de la ressource en ayant vérifié, au préalable, son identité ;
- le **endpoint de jeton** : permet d'échanger l'autorisation contre un jeton d'accès.

Le endpoint d'autorisation

Le endpoint d'autorisation est utilisé par les flux « code d'autorisation » et « octroi implicite ». Le client informe le serveur d'autorisation du flux d'autorisation désiré en mentionnant s'il souhaite réceptionner un code d'autorisation, un jeton d'accès ou une valeur d'extension enregistrée.

Après avoir achevé sa transaction avec le propriétaire de la ressource, le serveur d'autorisation redirige le client via son user-agent vers les endpoints de redirection du propriétaire de la ressource établis précédemment avec le serveur d'autorisation.

Le serveur d'autorisation doit alors comparer et retrouver au moins une valeur d'URI correspondante dans celles enregistrées. Si le serveur d'autorisation permet au client de changer dynamiquement les composants de la requête, le client doit s'assurer que la manipulation de cette dernière par une personne malintentionnée ne peut conduire à un « abus de redirection vers les endpoints ».

Si une requête d'autorisation échoue à la validation, le serveur d'autorisation doit informer le propriétaire de la ressource concernant l'erreur et il ne doit pas systématiquement rediriger le user-agent vers l'URI de redirection. La redirection ne doit également pas s'effectuer si les URI ne sont pas enregistrés ou s'ils ne sont pas fiables. Ceci afin d'éviter que le endpoint d'autorisation ne soit utilisé comme un « redirecteur ouvert » (redirecteur pouvant effectuer des redirections vers n'importe quel domaine).

Il est primordial que le client s'assure que ses propres scripts, utilisés pour extraire et retirer les identifiants, soient exécutés en premier lieu lorsque la réponse contient immédiatement l'URI de redirection afin d'éviter toute exécution de script non souhaitée pouvant compromettre la sécurité.

Le endpoint du jeton

Le endpoint du jeton est utilisé par le client (sauf dans le flux d'octroi implicite) afin d'obtenir un jeton d'accès en présentant son autorisation ou son jeton d'actualisation.

Les clients, effectuant des requêtes vers le endpoint du jeton, doivent s'être authentifiés au préalable avec le serveur d'autorisation :

Cette authentification du client permet de :

- renforcer la liaison des jetons d'actualisation et des codes d'autorisation avec les clients qui les ont fournis ;
- récupérer un client compromis en désactivant ses jetons d'actualisation ou en changeant ses identifiants (plus rapide que de révoquer un ensemble entier de jetons d'actualisation) ;

- implémenter les bonnes pratiques de la gestion de l'authentification nécessitant une rotation périodique des identifiants (plus simple qu'avec les jetons d'actualisation).

Champ du jeton d'accès

Les endpoints du jeton et d'autorisation permettent au client de spécifier le champ d'application de la requête d'accès. A son tour, le serveur d'autorisation informe le client sur le champ d'application que le jeton d'accès couvre.

Le serveur d'autorisation peut ignorer le champ d'application demandé par le client. Si le champ du serveur et du client sont différents, le serveur d'autorisation devrait informer le client du champ d'application qui a été effectivement octroyé.

3.3.4.4 Les différents flux d'autorisation

3.3.4.4.1 Le flux « code d'autorisation »

Le flux « code d'autorisation », aussi appelé « authorization code » dans la littérature, est représenté à la Figure 3-3. Il est utilisé pour obtenir les jetons d'accès et d'actualisation. Il est optimisé pour la confidentialité avec les clients.

En effet, le code d'autorisation est obtenu en utilisant un serveur d'autorisation comme intermédiaire entre le client et le propriétaire de la ressource. Le client redirige le propriétaire de la ressource vers un serveur d'autorisation qui, à son tour, redirige le propriétaire de la ressource vers le client avec le code d'autorisation. Le serveur d'autorisation aura, au préalable, authentifié le propriétaire de la ressource et obtenu l'autorisation appropriée.

Avantages:

- authentification du client ;
- les identifiants des propriétaires des ressources ne sont jamais partagés avec le client mais uniquement avec le serveur d'autorisation ;
- transmission des jetons d'accès directement au client (pas d'exposition aux autres acteurs).

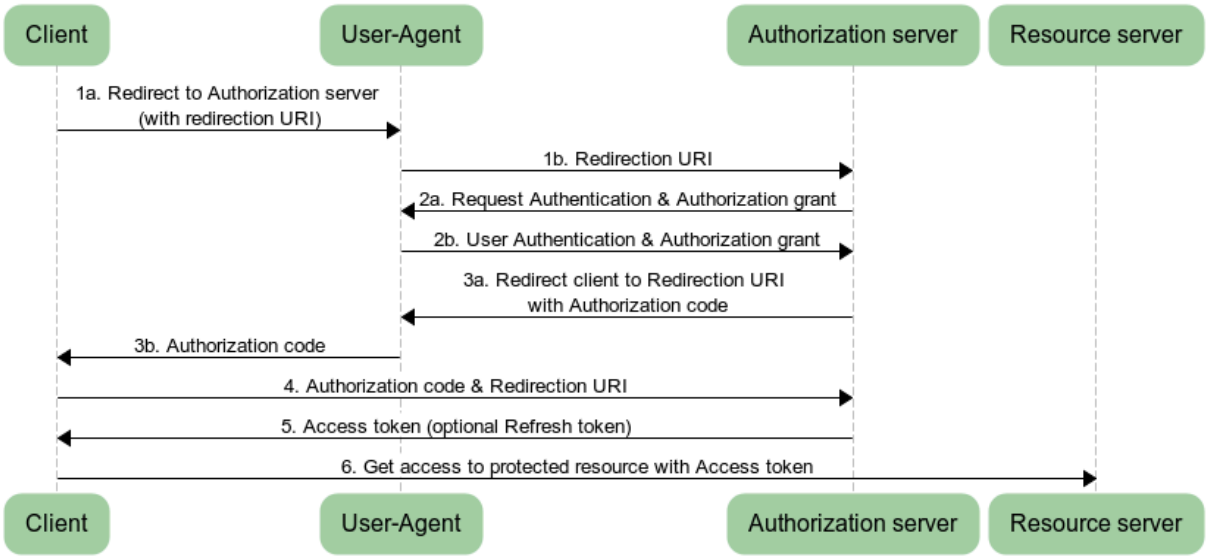


Figure 3-3 : Flux du code d'autorisation OAuth v2.0

1. Le client initie la connexion en redirigeant le user-agent de l'utilisateur vers le endpoint d'autorisation. Le client ajoute son identifiant de client, le champ

d'application demandé, l'état local ainsi que l'URI de redirection vers lequel le serveur d'autorisation renverra le user-agent après que l'accès ait été octroyé. Le serveur d'autorisation valide la requête pour s'assurer que tous les paramètres requis sont présents et valides.

2. Le serveur d'autorisation authentifie le propriétaire de la ressource et établit le lien, et ce, que le propriétaire de la ressource ait autorisé ou non la requête d'accès du client.
3. Le serveur d'autorisation fournit un code d'autorisation et le délivre au client (en utilisant l'URI de redirection défini à l'étape 1). Si la requête échoue, le serveur d'autorisation devrait en informer le propriétaire de la ressource et ne devrait pas automatiquement rediriger le user-agent vers l'URI vu la présence d'un message d'erreur.
4. Le client demande un jeton d'accès en incluant le code d'autorisation reçu à l'étape précédente. Le client inclut l'URI de redirection pour obtenir le code d'autorisation.
   Si le type de client est confidentiel ou que le client s'est vu assigner des exigences d'authentification, le client doit s'authentifier avec le serveur d'autorisation.
5. Le serveur d'autorisation authentifie le client, valide le code d'autorisation, et s'assure que l'URI de redirection reçu correspond bien à l'URI utilisé pour rediriger le client. Si l'URI est valide, le serveur d'autorisation fournit un jeton d'accès et éventuellement un jeton d'actualisation. Si l'authentification du client échoue, le serveur d'autorisation retourne une erreur.
6. Le client est maintenant en possession du jeton d'accès et peut, par ce biais, accéder aux ressources protégées.

3.3.4.4.2 Le flux « octroi implicite »

Le type « octroi implicite », aussi appelé « implicit grant » dans la littérature, est une version simplifiée du flux « code d'autorisation » optimisé pour les clients implémentés dans un navigateur utilisant un langage de script tel que JavaScript. Ce type de flux, représenté à la Figure 3-4, est utilisé pour obtenir des jetons d'accès (la fourniture de jetons d'actualisation n'est pas supportée) et est optimisé pour des clients publics, connus pour effectuer des redirections URI particulières.

Dans ce flux, le client reçoit immédiatement un jeton d'accès directement après avoir reçu l'autorisation du propriétaire de la ressource. Ce flux n'inclut pas l'authentification client, et dépend de la présence du propriétaire de la ressource et de l'enregistrement de l'URI de redirection. Comme le jeton d'accès est encodé dans l'URI de redirection, il peut être exposé au propriétaire de la ressource et, donc à d'autres services résidant sur le périphérique.

Avantages :

- améliore la réactivité et l'efficacité des clients puisqu'il réduit le nombre d'aller-retour requis pour obtenir un jeton d'accès.

Inconvénients (implications en termes de sécurité):

- aucune authentification client dans certains cas (dans d'autres cas, l'URI de redirection pour délivrer le jeton d'accès au client peut être utilisé) ;
- le jeton d'accès peut être exposé au propriétaire de la ressource mais aussi à tout intervenant ayant accès au user-agent du propriétaire de la ressource.

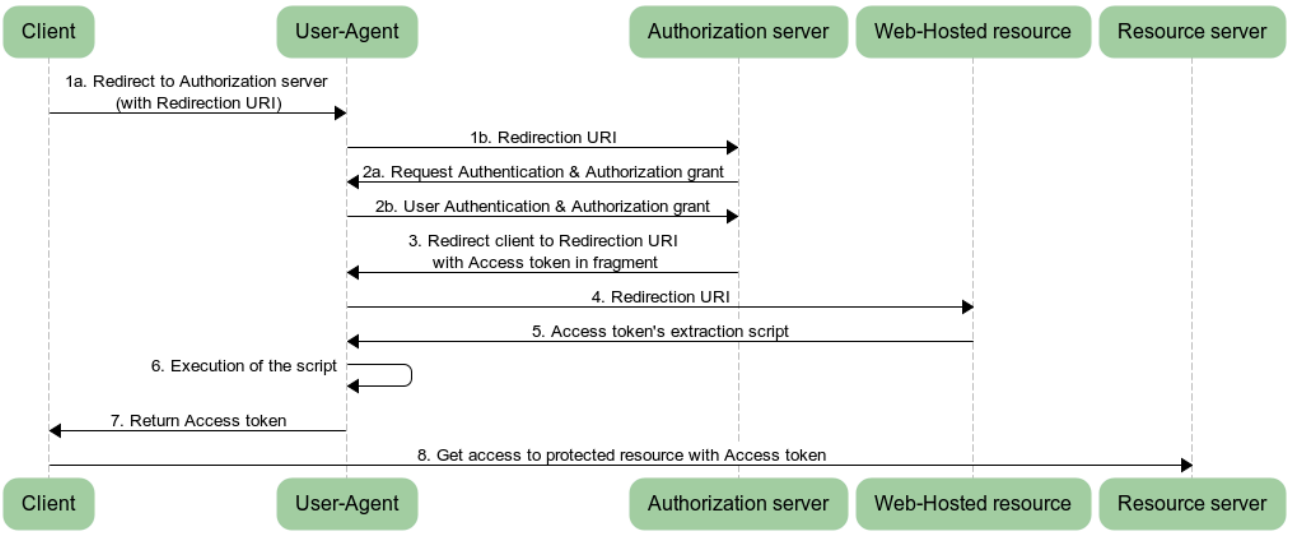


Figure 3-4 : Flux d'octroi implicite OAuth v2.0

1. Le client initie la connexion en redirigeant le user-agent de l'utilisateur vers le endpoint d'autorisation. Le client ajoute son identifiant de client, le champ d'application demandé, l'état local ainsi que l'URI de redirection vers lequel le serveur d'autorisation renverra le user-agent après que l'accès ait été octroyé. Le serveur d'autorisation valide la requête pour s'assurer que tous les paramètres requis sont présents et valides.
   Les seules différences vis-à-vis du flux « code d'autorisation » concernent:
   a. la valeur du paramètre response_type ;
   b. le serveur d'autorisation doit vérifier que l'URI de redirection vers lequel il redirigera le jeton d'accès correspond bien à l'URI de redirection enregistré par le client.
2. Le serveur d'autorisation authentifie le propriétaire de la ressource et établit le lien, et ce, que le propriétaire de la ressource ait autorisé ou non la requête d'accès du client.
3. Le serveur d'autorisation redirige le user-agent vers le client en utilisant l'URI de redirection fourni plus tôt (dans la requête ou durant l'enregistrement du client) et délivre uniquement un jeton d'accès au client. Si la requête échoue, le serveur d'autorisation devrait en informer le propriétaire de la ressource et ne devrait pas automatiquement rediriger le user-agent vers l'URI de redirection invalide.
4. Le user-agent suit les instructions de redirection en faisant une requête à la ressource cliente hébergée sur le web.
5. La ressource cliente hébergée par le web renvoie un script capable d'accéder à l'URI de redirection, et pouvant extraire le jeton d'accès et tout autre paramètre fourni.
6. Le user-agent exécute localement le script reçu pour extraire le jeton d'accès.
7. Le user-agent passe le jeton d'accès au client.
8. Le client est maintenant en possession du jeton d'accès et peut, par ce biais, accéder aux ressources protégées.

3.3.4.4.3 Le flux « identifiants par mot de passe du propriétaire de la ressource »

Le type d'autorisation des identifiants par mot de passe du propriétaire de la ressource, aussi appelé « ressource owner password credentials » dans la littérature et représenté à la Figure 3-5, est souhaitable si le propriétaire de la ressource a établi une relation de confiance avec le client (exemple : système d'exploitation du périphérique ou une application hautement privilégiée). Le serveur d'autorisation devrait être attentif

lorsqu'il active ce type de flux, et ne devrait l'autoriser que si les autres flux ne sont pas envisageables. Ce flux est généralement utilisé pour les clients capables d'obtenir les identifiants du propriétaire de la ressource (nom d'utilisateur et mot de passe) à l'aide d'un formulaire interactif. Ces identifiants seront échangés contre un jeton. Ce type de flux permet également de migrer des clients existants.
Avantage :

- élimine la nécessité pour le client de stocker les identifiants pour une utilisation future en échangeant les identifiants avec un jeton d'accès de longue durée ou un jeton d'actualisation.

Inconvénient (implications en termes de sécurité) :

- nécessite une confiance extrême entre le propriétaire de la ressource et le client.

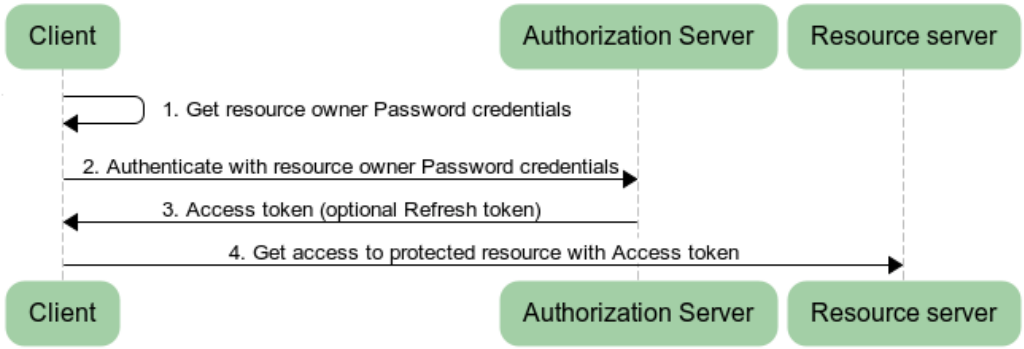


Figure 3-5 : Flux des identifiants par mot de passe du propriétaire de la ressource OAuth v2.0

1. Le propriétaire de la ressource transfère ses identifiants (nom d'utilisateur et mot de passe) au client.
2. Le client demande un jeton d'accès au endpoint de même nom à l'aide des identifiants reçus du propriétaire de la ressource.
   Si la requête du jeton d'accès utilise le mot de passe du propriétaire de la ressource, le serveur d'autorisation doit protéger le endpoint contre tout type d'attaque par force brute.
3. Le serveur d'autorisation authentifie le client, valide les identifiants du propriétaire de la ressource et fournit un jeton d'accès et, éventuellement, un jeton d'actualisation. Si la requête échoue à l'authentification du client ou si l'authentification est invalide, le serveur d'autorisation retournera une erreur.
4. Le client est maintenant en possession du jeton d'accès et peut, par ce biais, accéder aux ressources protégées.

3.3.4.4.4 Le flux « identifiants du client »

Le type de flux « identifiants du client, aussi appelé « client credentials grant » dans la littérature, est représenté à la Figure 3-6. Dans ce flux, le client peut demander un jeton d'accès en utilisant uniquement ses identifiants du client. Ce type de flux est typiquement utilisé lorsque le client fait une demande d'accès aux ressources protégées sous son contrôle, ou celles d'un autre propriétaire ayant un partenariat avec le serveur d'autorisation. Ce type de flux doit être utilisé par des clients confidentiels.

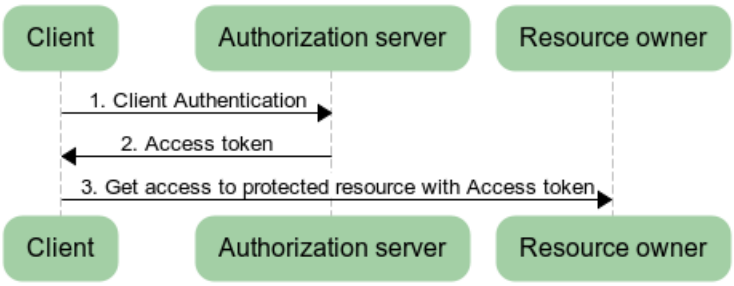


Figure 3-6 : Flux identifiants du client OAuth v2.0

1. Le client s'authentifie auprès du serveur d'autorisation et demande un jeton d'accès. Pour ce faire, il exécute une requête sur le endpoint du jeton.
2. Si la requête du jeton d'accès est valide et autorisée, le serveur d'autorisation fournira un jeton d'accès (un jeton d'actualisation ne devrait pas être inclus). Si la demande d'authentification client échoue ou est invalide, le serveur d'autorisation renverra un message d'erreur.
3. Le client est maintenant en possession du jeton d'accès et peut, par ce biais, accéder aux ressources protégées.

3.3.4.4.5 Le flux « étendu »

Le client peut utiliser un type de flux étendu. Le serveur d'autorisation définit lui-même la façon dont les jetons sont fournis et force le client à s'adapter à ses exigences.

Si la requête du jeton d'accès est valide et autorisée, le serveur d'autorisation fournira un jeton et, facultativement, un jeton d'actualisation. Si la demande échoue lors de l'authentification client ou qu'elle n'est pas valide, le serveur d'autorisation renverra une erreur.

Tous ces types de flux étendu sont décrits en détail dans des spécifications séparées.

3.3.4.5 Fournir un jeton d'accès, actualiser un jeton d'accès & accéder aux ressources

Le flux de la Figure 3-7 représente les interactions pour demander un jeton d'accès, actualiser un jeton d'accès à l'aide d'un jeton d'actualisation et finalement accéder aux ressources.

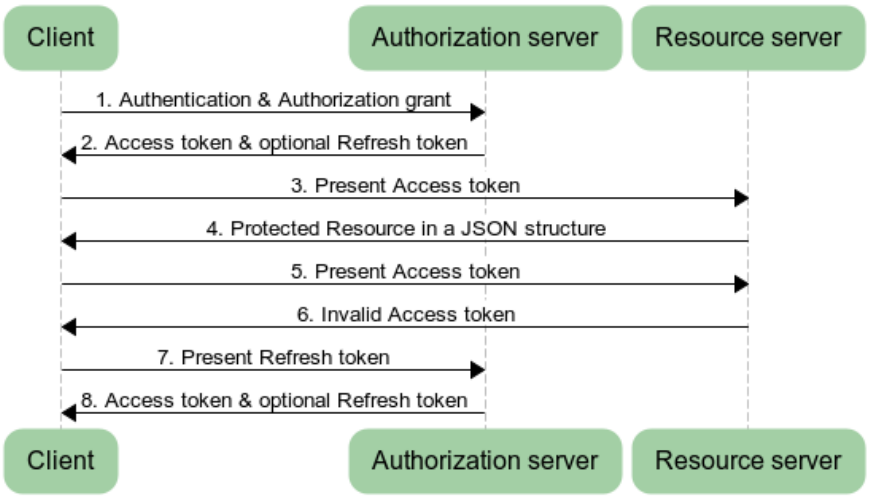


Figure 3-7 : Actualiser un jeton d'accès expiré OAuth v2.0

1. Le client demande un jeton d'accès en s'authentifiant auprès du serveur d'autorisation et en lui présentant l'autorisation.

2. Le serveur d'autorisation authentifie le client et valide l'autorisation.
   Il fournit ensuite un jeton d'accès et un jeton d'actualisation ou renvoie une erreur (si l'authentification échoue ou si l'autorisation n'est pas valide).
3. Le client effectue une demande d'accès à une ressource protégée sur le serveur correspondant en présentant son jeton d'accès. Ce serveur doit valider le jeton d'accès, s'assurer qu'il n'est pas expiré et que son champ d'application couvre la ressource demandée.
4. Le serveur valide le jeton et répond à la demande. Les paramètres de la réponse sont inclus dans le corps de la réponse et sérialisés dans une structure JSON.
5. Les deux étapes précédentes sont répétées tant que le jeton est valide. Si le client sait que son jeton est expiré il passe à l'étape 7.
6. Si le jeton n'est plus valide une erreur est retournée.
7. Le client demande un nouveau jeton d'accès en s'authentifiant avec le serveur d'autorisation et en présentant le jeton d'actualisation. Il effectue la requête vers le endpoint du jeton. Si le client est confidentiel ou s'il possède des identifiants, il doit s'authentifier avec le serveur d'autorisation.
8. Si la validation s'opère correctement et si la requête est autorisée, le serveur d'autorisation fournit un jeton d'accès et éventuellement un nouveau jeton d'actualisation. Dans ce cas, le client et le serveur d'autorisation doivent se débarrasser de l'ancien jeton et le remplacer par le nouveau. Son champ d'application doit être identique à celui inclus par le client dans la requête.

3.3.5 Considérations en termes de sécurité

OAuth v2 étant un framework flexible et extensible, voici cependant un ensemble de points d'attention en termes de sécurité :

Authentification client :

- le serveur d'autorisation est encouragé à utiliser des authentifications fortes avec le client. Les applications clientes web doivent assurer la confidentialité de tout type d'identifiants ;
- le serveur d'autorisation ne doit pas fournir les identifiants du client à une application native ou à une application basée sur un user-agent pour l'authentification client ;
- une identification du client est plus que conseillée. Si elle n'est pas possible par les moyens habituels, le serveur d'autorisation devrait trouver un autre moyen de valider l'identité du client (en demandant une confirmation au propriétaire de la ressource et en exigeant l'enregistrement de l'URI de redirection).

Le serveur d'authentification doit être conscient qu'une communication avec des clients non authentifiés peut être dangereuse et il doit prendre les dispositions nécessaires à cet égard.

Emprunt d'identité client :

- les identifiants du clients doivent être gardés confidentiels ;
- le serveur d'autorisation doit identifier le client dès que possible. Autrement, il prendra les précautions nécessaires pour protéger les propriétaires de la ressource des clients malintentionnés (en demandant au propriétaire de confirmer l'identité de ce client) ;
- le serveur d'autorisation devrait renforcer l'authentification du propriétaire de la ressource et lui fournir toutes les informations concernant le client, les caractéristiques du jeton,… ;
- le serveur d'autorisation ne doit pas traiter les requêtes répétées automatiquement sans une interaction avec le propriétaire de la ressource et sans s'être assuré de l'identité du client à chaque requête.

Jetons d'accès :

- les jetons d'accès doivent être gardés confidentiels et partagés uniquement avec le serveur d'autorisation, le serveur de la ressource concernée et le client qui les a reçus ;
- le serveur d'autorisation doit s'assurer que les jetons d'accès ne peuvent pas être générés, modifiés, ou devinés pour reproduire un jeton d'accès valide par des tiers non autorisés (principalement dans le type d'octroi implicite ou le jeton est transmis dans l'URI) ;
- le client devrait demander le jeton d'accès avec un champ d'application et une durée de vie minimale. Dans cette optique le serveur pourrait prendre en compte l'identité du client afin de limiter le champ d'application vis-à-vis du champ demandé.

Jetons d'actualisation:

- les jetons d'actualisation doivent être gardés confidentiels et partagés uniquement avec le serveur d'autorisation et le client qui les a reçus. Le serveur d'autorisation doit maintenir un lien entre le jeton d'actualisation et le client. Ce lien doit être vérifié même si l'identité du client a été authentifiée. Si l'authentification du client n'est pas possible, le serveur d''autorisation doit déployer d'autres moyens pour détecter l'abus d'utilisation de jetons d'actualisation ;
- le serveur d'autorisation doit s'assurer que les jetons d'actualisation ne peuvent pas être générés, modifiés, ou devinés pour produire un jeton d'actualisation valide par des tiers non autorisés.

Codes d'autorisation :

- des efforts devraient être réalisés pour garder la confidentialité des codes d'autorisation: utilisation de SSL et TLS si l'URI de redirection est utilisé. Les codes d'autorisation sont transmis par des redirections de user-agent et pourraient donc être divulgués ;
- si le client dépend du code d'autorisation pour sa propre authentification en tant que propriétaire de ressource, le endpoint de redirection client doit exiger TLS ;
- les codes d'autorisation doivent être de courte durée et à utilisation unique. S'il s'avérait qu'un code d'autorisation était utilisé plusieurs fois, le serveur devrait révoquer l'ensemble des jetons déjà autorisés sur base de ce code ;
- les serveurs d'autorisation doivent authentifier le client si c'est possible et s'assurer que le code d'autorisation a été délivré au même client.

Hameçonnage (flux « code d'autorisation ») :

- lors de l'utilisation du flux « code d'autorisation », il faut s'assurer que l'URI de redirection n'a pas été manipulé afin de rediriger le propriétaire de la ressource vers un URI sous le contrôle d'une personne malintentionnée. Dans ce cas elle jouerait l'intermédiaire entre le propriétaire de la ressource et les différentes parties. Pour se protéger contre ce type de pratique, le serveur d'autorisation doit s'assurer que l'URI de redirection utilisé pour obtenir le code d'autorisation est identique à l'URI fourni lorsque le code d'autorisation est échangé contre un jeton d'accès. Le serveur d'autorisation doit exiger des clients publics (et devrait exiger des clients confidentiels) l'enregistrement de l'URI de redirection. Si une valeur est fournie dans l'URI, le serveur d'autorisation doit la valider en vis-à-vis des valeurs enregistrées.

Flux « Identifiants de mot de passe du propriétaire de la ressource » :

- ce type d'autorisation est souvent utilisé pour l'héritage et la migration. Il réduit le risque du stockage du nom d'utilisateur et mot de passe par le client mais n'élimine pas le besoin d'exposer des identifiants au client. Le client pourrait utiliser de façon abusive le mot de passe et ce dernier pourrait, donc, être révélé à un attaquant ;
- le propriétaire de la ressource n'a aucun contrôle sur le processus d'autorisation, le client peut obtenir des jetons d'accès avec un champ d'application et une durée de vie plus large que ceux désirés par le propriétaire de la ressource. Pour ces raisons, le serveur d'autorisation et les clients devraient éviter l'utilisation de ce type de flux et en utiliser un autre si c'est envisageable.

Requête de confidentialité :

- les jetons d'accès, les jetons d'actualisation, les mots de passe du propriétaire de la ressource ainsi que les identifiants du client ne doivent jamais être transmis en clair. Les codes d'autorisation ne devraient pas l'être non plus.

Authenticité des endpoints :

- afin d'éviter les attaques de l'homme du milieu et l'hameçonnage, le serveur d'autorisation doit implémenter et exiger TLS avec le serveur d'authentification pour toutes les requêtes envoyées aux endpoints du jeton et d'autorisation. Le client doit valider le certificat du serveur d'autorisation. Il doit également valider la liaison du serveur à son nom de domaine.

Attaques de découverte des identifiants :

- le serveur d'autorisation doit se protéger des pirates qui tenteraient de deviner les jetons d'accès, les codes d'autorisation, les jetons d'actualisation, les mots de passe du propriétaire de la ressource et les identifiants du client. Pour tout type d'identifiant fourni à un tiers autre que les utilisateurs, le serveur d'autorisation doit utiliser un niveau raisonnable d'entropie afin d'atténuer ce type d'attaque. Le serveur d'autorisation doit utiliser d'autres moyens pour protéger les identifiants destinés aux utilisateurs.

Attaques par hameçonnage :

- l'utilisateur étant très souvent redirigé, il peut devenir habitué à ce type de pratique et peut donc ne plus s'en méfier. Un pirate pourrait tenter de voler les identifiants du propriétaire de la ressource. Afin de pallier à ce problème, les fournisseurs devraient éduquer les utilisateurs à se méfier de ce type d'attaque et devraient fournir des mécanismes qui facilitent la confirmation par les utilisateurs de l'authenticité de leur site et du serveur d'autorisation.

Attaques Cross-Site Request Forgery :

- le client doit mettre en place, sur son URI de redirection, une protection contre les attaques de type CSRF. Le paramètre *state* inclus dans chacune des requêtes envoyées vers le endpoint de redirection va pallier à ce problème. Ce paramètre lie la requête à l'état d'authentification du user-agent. La valeur de celui-ci permet au client de vérifier la validité de la requête en la comparant avec l'état authentifié du client. Ce paramètre *state* contient une valeur qui ne peut être devinée. L'état authentifié du user-agent doit être conservé à un endroit accessible uniquement au client et au user-agent ;
- le serveur d'autorisation doit implémenter une protection CSRF pour son endpoint d'autorisation. Il doit, également, s'assurer qu'un client malintentionné

ne peut obtenir une autorisation sans le consentement explicite et la sensibilisation du propriétaire de la ressource.

Clickjacking :

- pour prévenir ce type d'attaque, les applications natives devraient utiliser des navigateurs externes lorsqu'elles demandent une autorisation à l'utilisateur. L'utilisation de navigateurs embarqués dans un iframe est déconseillée. La plupart des nouveaux navigateurs permettent d'éviter ce genre de pratique grâce à l'en-tête *x-frame-options*. Les deux valeurs possibles de cet en-tête sont *deny* ou *sameorigin*, permettant de bloquer, respectivement, soit tous les types de frames, soit celles n'ayant pas la même origine.

Injection de code et validation des entrées :

- le serveur d'autorisation et le client doivent valider toute valeur reçue et plus particulièrement les paramètres *state* et *redirect_uri.* Ceci pour des raisons évidentes évoquées dans les points d'attention liés :
    - aux attaques Cross-Site Request Forgery ;
    - à l'hameçonnage ;
    - à la manipulation de l'URI de redirection.

Redirecteurs ouverts :

- les endpoints de redirection et d'autorisation du serveur peuvent être mal configurés et opérés comme un « redirecteur ouvert » (endpoint utilisant un paramètre pour rediriger automatiquement un user-agent à l'emplacement indiqué par la valeur du paramètre sans aucune validation) ;
- les redirecteurs ouverts peuvent être utilisés dans les attaques par hameçonnage. Ils peuvent également l'être par un attaquant pour amener les utilisateurs à visiter un URI qui ressemble à une destination familière et de confiance. Si le serveur d'autorisation permet au client d'enregistrer une partie de l'URI de redirection, un tiers malintentionné peut utiliser un redirecteur ouvert exploité par des clients pour construire un URI de redirection qui passera la validation du serveur d'autorisation mais qui enverra le code d'autorisation ou le jeton d'accès au endpoint sous le contrôle de l'attaquant.

3.3.6 Failles de sécurité

Si les différentes recommandations et exigences du protocole OAuth v2 ne sont pas respectées, et donc, si les considérations en termes de sécurité reprises ci-dessus ne sont pas suivies à la lettre, les différents tiers et leurs communications s'exposent naturellement à certains risques qui ne doivent pas être pris à la légère. Les conséquences peuvent être dramatiques. Le but de cette section est de sensibiliser à la sécurité toute personne souhaitant implémenter OAuth v2.

Cette section n'a pas pour objectif d'énumérer les dangers auxquels s'exposent les développeurs. Cette analyse des risques ayant déjà été faite par ailleurs [Lodderstedt et al, 2011]. Pour chacun des flux, ce document décrit les attaques, leurs impacts possibles ainsi que les précautions à prendre afin de s'en prévenir et de s'en protéger. La seconde partie du document énumère les différents aspects du protocole, donne des recommandations en vis-à-vis et mentionne contre quelles attaques ces recommandations vous protègent. La majorité des points d'attention ont été repris dans la section précédente. Ce document possède néanmoins d'autres points d'attention liés à la sécurité du poste client.

Une analyse d'une implémentation de OAuth v2 est très facilement réalisable, en parcourant le document « draft-ietf-oauth-v2-threatmodel-01 » [Lodderstedt et al,

2011] et en ayant au préalable, enregistré le flux HTTP de l'authentification avec un outil tel que « Live HTTP headers » (plugin[1] pour Firefox). La comparaison entre ce document et la capture, ainsi que différents petits tests vous donneront une très bonne base pour étudier la fiabilité de l'implémentation en termes de sécurité. Pour compléter cette analyse, il faudrait pouvoir se positionner entre chacun des acteurs afin de capturer le flux de l'échange et avoir accès à chacun des serveurs agissant en tant qu'acteurs.

Aucune faiblesse du protocole n'est actuellement relevée si les exigences et recommandations sont suivies à la lettre et implémentées telles que décrites dans la spécification. Il est, par contre, dommage que la spécification de OAuth 2.0 ne soit pas plus stricte en matière de sécurité des communications et que certaines recommandations en matière de sécurité n'aient pas été reprises en tant qu'exigences.

3.3.7 Synthèse

La spécification de OAuth 2.0 offre plusieurs possibilités pour obtenir le jeton d'accès permettant la délégation des autorisations. Ces fonctionnalités permettent au protocole d'être plus souple vis-à-vis des différents cas d'utilisation. La mise en place du flux « code d'autorisation » est la plus sûre et celle qui est recommandée.

La spécification se veut largement plus détaillée et plus simple à mettre en place. Elle fournit des conseils éclairés sur la façon d'implémenter ce protocole de façon sécurisée. Le document n'est cependant pas très lisible tant l'information et les conseils sont nombreux.

Notons les différents points suivants sur lesquels la spécification insiste :

- aucune hypothèse ne doit être faite sur de quelconques variables ;
- tout type de requête contenant des données sensibles ou des textes en clair devrait exiger un mécanisme de sécurité de la couche transport d'autant plus si le transport se fait sur un réseau ouvert. Le manque d'une telle sécurité au niveau de la couche transport peut avoir un impact important sur la sécurité du client et les ressources protégées pour lesquelles l'accès a été autorisé.

Le développeur est donc averti des risques mais n'est pas toujours obligé d'implémenter une couche de sécurité telle que TLS ou SSL, ce qui semble souvent être critiqué par la communauté d'utilisateurs OAuth.

Le principal point fort de OAuth réside donc dans la délégation d'autorisation, quelque soit le cas d'utilisation, et surtout peu importe les acteurs impliqués.

[1] https://addons.mozilla.org/fr/firefox/addon/live-http-headers/

4 Facebook Connect

4.1 Introduction

Facebook, la plate-forme de réseau social créée en février 2004 par Mark Zuckerberg, offre la possibilité d'utiliser, d'intégrer et d'enrichir le réseau social par l'intermédiaire de plugins. Du simple bouton « J'aime », qui publiera automatiquement vos préférences sur le réseau social, à des modules avancés permettant aux développeurs d'accéder aux données de l'utilisateur, la gamme de plugins[1] est très riche et chacun de ceux-ci fait partie d'un service pouvant être offert à la communauté des développeurs.

Parmi ces services, on retrouve Facebook Connect. Il a pour objectif d'authentifier et de fournir un système d'autorisation par l'intermédiaire de Facebook jouant ainsi le rôle d'un fournisseur d'identité.

Ces données et les services offerts par la plate-forme Facebook sont repris dans la **Figure** 4-1.

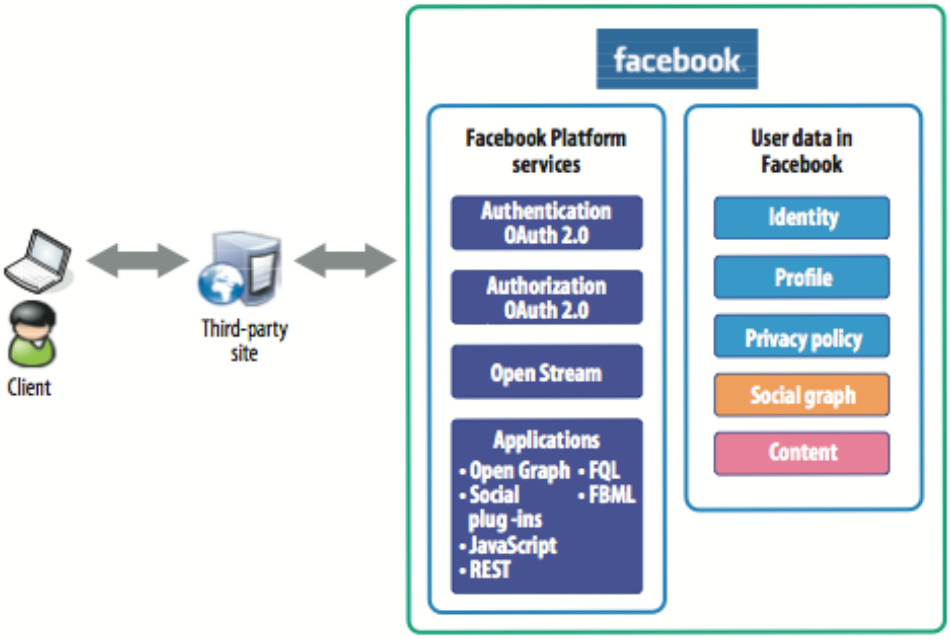


Figure 4-1 : Données et services de la plate-forme Facebook [Nam Ko, 2010]

L'objet de cette section portera principalement sur l'aspect technique de l'authentification et des autorisations. Nous aborderons également l'atteinte à la vie privée des utilisateurs (aspect critique pour les fournisseurs d'identité dans les réseaux sociaux) et la façon dont l'utilisateur en est informé.

L'étude de ce protocole a principalement été réalisée durant les mois de novembre et décembre 2011.

4.2 Facebook Connect

Facebook Connect est une plate-forme d'authentification unique permettant à un utilisateur de pouvoir se connecter à un site web (appelé généralement *application* sur Facebook) par l'intermédiaire de son identité Facebook.

Cette plate-forme lancée en décembre 2008 possède déjà plusieurs milliers de partenaires et détient, de surcroît à ses informations personnelles et à l'insu de

[1] Les plugins sont disponibles à l'adresse http://developers.Facebook.com/docs/plugins/

l'utilisateur, ses habitudes de navigation. Fort de près de 900 millions d'utilisateurs, Facebook est un important fournisseur d'identité. Ce chiffre croît actuellement à raison de 150 millions d'utilisateurs tous les 6 mois dans le monde [Wikipedia, 2012a].

Le fournisseur de services intégrant Facebook Connect a également la possibilité de pouvoir consulter très facilement via le Graph API[1] bon nombre d'informations publiques mais également privées, moyennant une demande d'autorisation à l'utilisateur.

Ci-dessous, les principaux avantages et inconvénients que présente ce module sont classés suivant le rôle des acteurs.

Le point de vue des développeurs :

Avantages :

- tirer parti de plus de 900 millions d'utilisateurs Facebook préenregistrés et ainsi avoir déjà un site pour lequel on a déjà des millions d'utilisateurs (pré)inscrits ;
- disposer d'un module d'authentification prêt à l'emploi permettant au développeur de se concentrer sur son business ;
- obtenir tout type d'informations (sur la personne elle-même, ses liens/connexions, les avoirs de la personne) et faciliter l'accès à ces informations (informations larges et sans nul doute très riches) ;
- obtenir des statistiques d'accès/d'analyse au site via le plugin authentification.

Inconvénients :

- aucune connaissance sur le mode de fonctionnement de l'authentification utilisé ;
- dépendance complète à Facebook concernant l'authentification et tous les services liés. Bien que très stable, un maillon est ajouté à la chaîne, fragilisant les accès aux services développés ;
- contraintes imposées par Facebook pour l'utilisation de Facebook Connect (bouton de déconnexion obligatoire,…).

Le point de vue des utilisateurs :

Avantages :

- simplification des démarches administratives de l'inscription ;
- authentification facilitée ;
- avantages de l'authentification unique.

Inconvénients :

- le fournisseur de services accède à certaines informations concernant l'utilisateur avec l'accord de celui-ci. Les demandes de permissions, relativement bien décrites, sont très fréquentes. Ce phénomène finit par lasser l'utilisateur qui autorisera machinalement ce type de demande afin d'obtenir l'accès au service demandé.

Le point de vue de Facebook :

L'avantage (et l'objectif recherché) est simple et commercial: connaître les habitudes de navigation des utilisateurs de Facebook et utiliser ces informations à certaines fins (publicitaires,…), et en faire une plus-value. Au plus les sites de réseaux sociaux ont d'informations sur leurs utilisateurs, au plus ils se distinguent des autres et ont une valeur ajoutée non négligeable.

Au vu des nombreux éléments de la vie privée que possède Facebook et de toutes ces informations échangées, une analyse de Facebook Connect semble plus que nécessaire.

[1] http://developers.facebook.com/docs/reference/api/

4.3 Type et composition

Facebook Connect est une implémentation du protocole OAuth v2. Depuis le 1er septembre 2011, l'ensemble des sites utilisant précédemment le module "Legacy Login" [Facebook, 2011a] de Facebook ont du migrer vers Facebook Connect v2 pour des raisons de sécurité (je cite « *le mécanisme d'héritage passait les informations d'authentification dans la chaîne de la requête URL. Si cette chaine était manipulée de façon incorrecte, elle pouvait être passée à un tiers par l'intermédiaire du navigateur* » [Facebook, 2011a]).

Bien qu'une solution de rechange ait été implémentée, la faille était néanmoins présente. Pour les développeurs peu consciencieux, celle-ci permettait, à une personne malintentionnée, de récupérer toutes les informations souhaitées concernant un utilisateur.

Facebook Connect définit 3 acteurs principaux:

- le **navigateur de l'utilisateur** comparable à l'**utilisateur** ;
- l'**application Facebook** représentant le **fournisseur de services** ;
- **Facebook** représentant le **fournisseur d'identité**.

Facebook Connect supporte 2 types de flux OAuth v2 [Facebook, 2011b]:

- le flux « code d'autorisation » intitulé « flux coté serveur » sur la plate-forme Facebook ;
- le flux « octroi implicite» intitulé « flux coté client » sur la plate-forme Facebook.

Ces deux flux sont décrits dans la section « Etude du protocole ».

Rappelons que le flux « code d'autorisation » est la méthode recommandée par la spécification de OAuth v2 et, dans cette optique, l'implémentation « coté serveur » de Facebook Connect devrait être plus sûre.

4.4 Concepts de base

Les concepts de base de Facebook Connect consistent à :

- authentifier l'utilisateur (l'utilisateur est bien celui qu'il prétend être) ;
- autoriser l'application à accéder aux informations de l'utilisateur (garder une emprise sur ce qui est transmis à l'application tierce) ;
- authentifier l'application (vérifier que l'on transmet les informations à l'application concernée et pas à une autre).

Une fois ces trois étapes réalisées, le serveur web recevra un jeton d'accès qui lui permettra d'obtenir les informations souhaitées. Ce jeton d'accès donne la permission au partenaire de consulter la base de données Facebook via le Graph API[1] (anciennement REST API devenue obsolète) comme illustré dans la Figure 4-2.

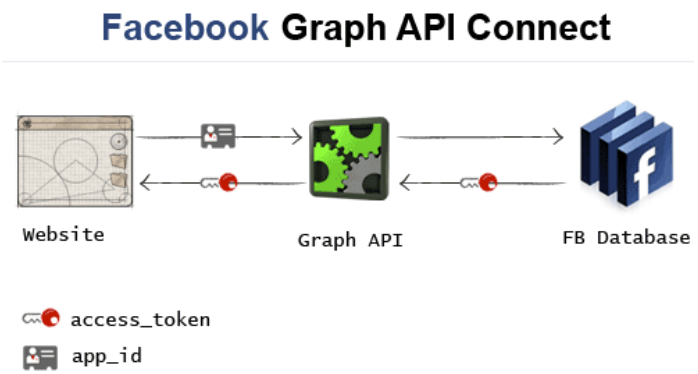


Figure 4-2 : Accès à la plate-forme Facebook via l'interface API [Tamada, 2011]

[1] http://developers.facebook.com/docs/reference/api/

4.5 Etude du protocole

Comme dit plus tôt, il existe deux types de flux d'authentification via Facebook [Facebook, 2011b]:

- le flux « coté client »: permet d'accéder aux fonctionnalités de l'API Graph, du FQL et de l'API REST ;
- le flux « coté serveur » : permet d'accéder aux fonctionnalités de l'API Graph, des dialogues, du rendu XFBML, du « canvas pages » et de l'API REST.

Au-delà des aspects performance et sécurité, le flux « coté serveur » sera utilisé lorsque le serveur web a besoin d'appeler le Graph API Facebook afin d'en retirer des informations. Dans le cas du flux « coté client » c'est le client (via son user-agent) qui effectuera ces appels. Exemple: si une application a besoin d'informations concernant un utilisateur lorsque celui-ci n'est pas connecté (cf. variable du champ d'application *offline_access*) le « flux coté serveur » sera utilisé. Ces deux mécanismes supportent OAuth v2.

Le choix du mécanisme d'authentification se fera, d'une part sur base de la complexité de l'application et l'objectif recherché par la plate-forme à développer et d'autre part sur la performance et la sécurité nécessaires et requises par le responsable de l'application.

4.5.1 Flux coté serveur

L'authentification et les autorisations

Les échanges de flux « coté serveur », de l'anglais « server-side flow », sont représentés par la Figure 4-3 et détaillés à la suite de celui-ci.

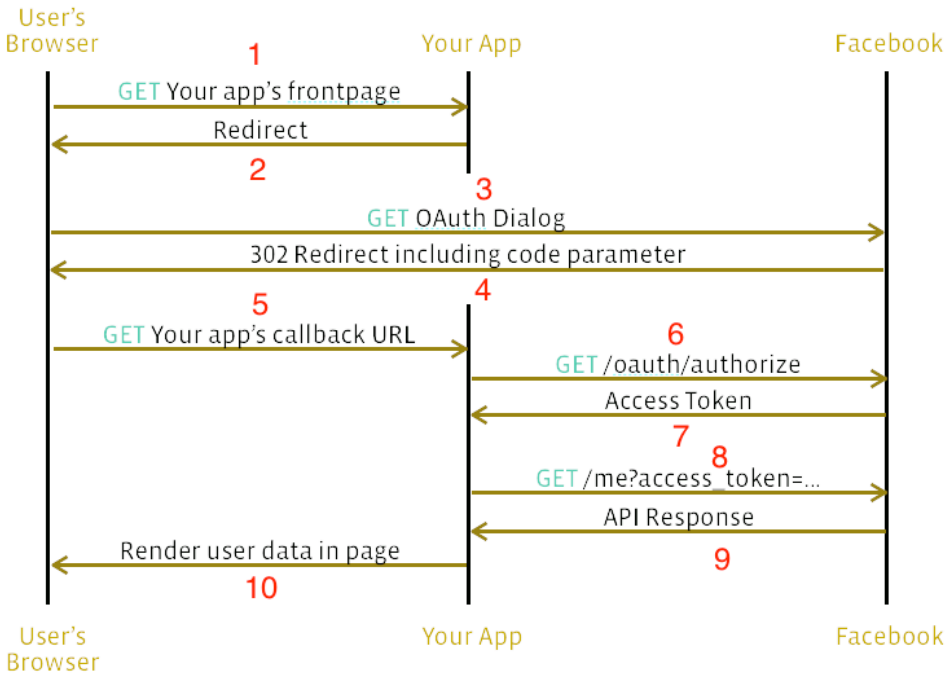


Figure 4-3 : Flux coté serveur de Facebook Connect [Facebook, 2011b]

1. L'utilisateur introduit l'adresse de l'application dans son navigateur et clique sur le bouton d'authentification Facebook.

2. L'utilisateur est redirigé vers le fournisseur d'identité afin d'obtenir les boîtes de Dialogue OAuth. Vous trouverez un aperçu de ces boîtes de dialogue à la Figure 4-4.

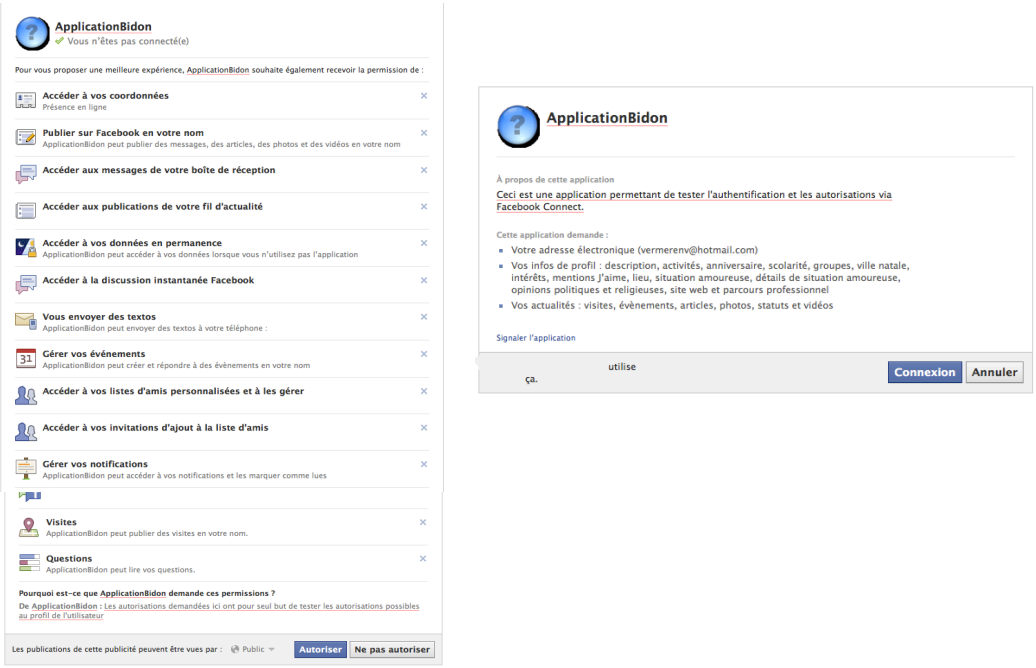


Figure 4-4 : Boîtes de dialogue OAuth de Facebook Connect

3. Requête GET pour authentifier l'utilisateur et obtenir les autorisations requises de celui-ci par l'intermédiaire des boîtes de dialogue OAuth. Les paramètres passés sont :
   - *client_id* : l'identifiant créé dans la « Developer App »[1] ;
   - *redirect_uri* : URL vers laquelle l'utilisateur sera redirigé une fois les autorisations validées ;
   - *scope* : ensemble des autorisations[2] d'accès au profil pour l'application cible.

   Les principaux arguments ont été mentionnés ci-dessus mais il en existe d'autres[3]. Si l'utilisateur est déjà connecté, Facebook validera le cookie de la session stocké sur la machine cliente.
4. La boîte de dialogue OAuth, si l'authentification s'est déroulée avec succès, redirigera l'utilisateur vers l'adresse introduite (cf. paramètre *redirect_uri*) avec :
   a. un code d'erreur, si l'autorisation a été refusée par l'utilisateur ;
   b. un code d'autorisation, si l'autorisation a été acceptée par l'utilisateur.
5. Redirection de l'utilisateur vers l'application pointée par l'adresse avec le code d'autorisation ou d'erreur.
6. Requête GET vers Facebook pour obtenir le jeton d'accès. Pour ce faire, les paramètres suivants sont nécessaires :
   - le code d'autorisation préalablement obtenu ;
   - l' « *app secret* » : le mot de passe obtenu dans la « Developer App » ;
   - l' « *app id* » : disponible également dans la « Developer App » ;
   - le paramètre *redirect_uri* utilisé à l'étape 3.

[1] https://developers.facebook.com/apps#_=_
[2] http://developers.facebook.com/docs/authentication/permissions
[3] http://developers.facebook.com/docs/reference/dialogs/oauth

7. Si les démarches précédentes se sont déroulées avec succès, Facebook renverra un jeton d'accès à l'utilisateur avec un délai d'expiration en paramètre de la réponse. Autrement, une erreur HTTP 400 sera renvoyée.
8. L'accès au profil de l'utilisateur et à ses données mentionnées dans le paramètre *scope* peut se faire en passant le jeton d'accès en paramètre.
9. Le Graph API Facebook répondra en fournissant une structure représentant les données de l'utilisateur dans une forme appropriée permettant de pouvoir facilement utiliser cette information. D'autres informations peuvent être obtenues dans un document disponible sur Facebook [Facebook, 2011c].
10. Il ne reste plus qu'à envoyer le résultat de la page à l'utilisateur.

Le flux 3 permet l'authentification de l'utilisateur et la validation des autorisations de l'application par celui-ci. Le flux 6 permet de vérifier l'authenticité du site en comparant le paramètre *redirect_uri* et l'URL du site configurée dans la « Developer App ».

Le logout

La déconnexion est relativement simple et consiste à appeler l'URL « https://www.facebook.com/logout.php ». La déconnexion ne sera effective qu'à condition de passer en paramètre le jeton d'accès obtenu en y ajoutant, si nécessaire, l'URL de redirection.

Cette déconnexion est donc faite à la fois au niveau du site web mais également au niveau de Facebook. On est donc par la même occasion déconnecté de l'intégralité des applications pour lesquelles on se serait authentifié par l'intermédiaire de Facebook. Ceci permet donc d'affirmer que l'application applique bien un mécanisme de déconnexion unique.

A noter, que l'autorisation de type « offline_access » permet au serveur d'avoir accès aux données de l'utilisateur même lorsque celui-ci est déconnecté. Ce jeton permet, à celui qui le possède, de donner un accès permanent aux données de l'utilisateur. Ce type de pratique est dangereux, d'autant plus que le jeton ne contient aucune information liée à la machine ayant reçu l'autorisation d'accéder aux données. Une attaque par rejeu sera donc facilement réalisable.

Si la variable « offline_access » n'est pas reprise dans la variable *scope*, l'accès aux données n'est effectivement plus valable une fois l'utilisateur déconnecté.

4.5.2 Flux coté client

L'authentification et les autorisations

Le flux « coté client », alias « client-side flow», est très semblable au flux « coté serveur ». La principale différence se situe au niveau des requêtes GET/POST qui sont initiées par le navigateur de l'utilisateur en lieu et place du serveur. Il se différencie surtout par une sécurité moins accrue mais en contrepartie par une meilleure réactivité et efficacité dans les communications.

Les échanges de flux sont représentés par la Figure 4-5 et détaillés à la suite de celui-ci.

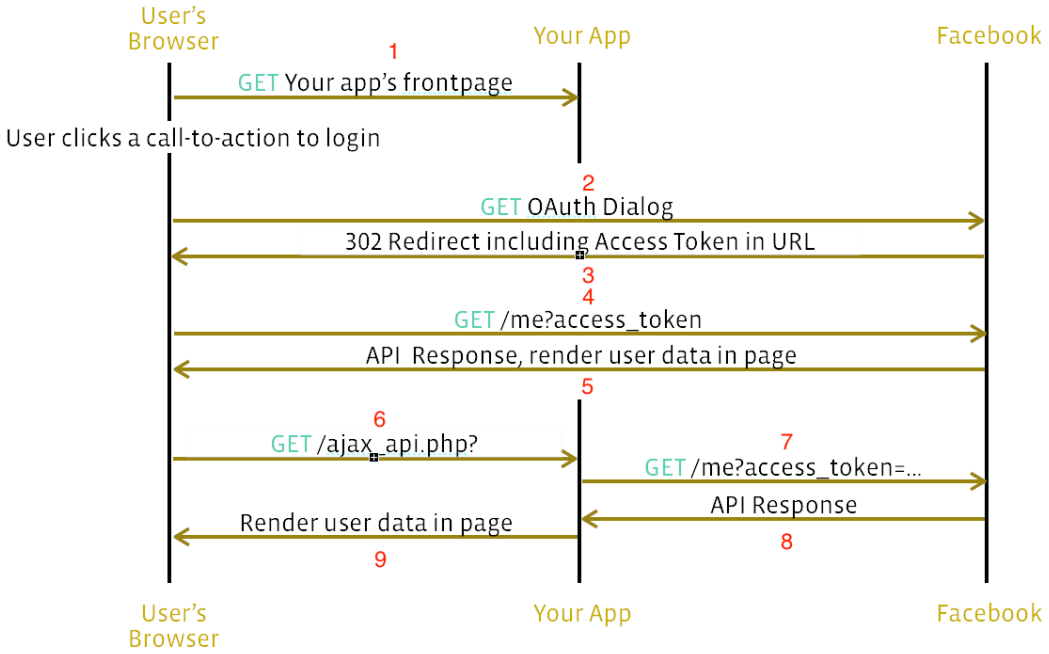


Figure 4-5 : Flux coté client de Facebook Connect [Facebook, 2011b]

1. L'utilisateur introduit l'adresse de l'application dans son navigateur et clique sur le bouton d'authentification Facebook.
2. Requête GET pour authentifier l'utilisateur et obtenir les autorisations requises de celui-ci par l'intermédiaire des boîtes de dialogue OAuth. Les paramètres passés sont :
   - *client_id* : l'identifiant créé dans la « Developer App »[1] ;
   - *redirect_uri* : URI vers lequel l'utilisateur sera redirigé une fois les autorisations acquises ;
   - *scope* : ensemble des autorisations[2] d'accès au profil pour l'application cible ;
   - *response_type* : type de réponse récupérée. Les valeurs possibles sont : *token* (pour recevoir un jeton), *code* (pour recevoir un code d'autorisation), *code token* (pour recevoir un jeton et un code d'autorisation).
3. La boîte de dialogue OAuth, si l'authentification s'est déroulée avec succès, redirigera l'utilisateur vers l'adresse introduite (cf. argument *redirect_uri*) avec :
   a. un code d'erreur, si l'autorisation a été refusée par l'utilisateur ;
   b. un jeton d'accès dans un fragment URI (uniquement accessible par le code coté client) si l'autorisation a été acceptée par l'utilisateur.
4. Redirection de l'utilisateur vers la page contenant les informations de l'utilisateur pour lesquelles il a explicitement donné son accord.
5. Retour des informations du Graph API de Facebook contenant les informations sur l'utilisateur.
6. Requête GET vers l'adresse concernée avec le jeton d'accès (obtenu précédemment par l'utilisateur) et l'expiration de celui-ci en paramètre.
7. L'application peut maintenant utiliser le jeton d'accès dans le but d'acquérir les informations sur l'utilisateur. C'est également durant cette étape que l'application s'authentifiera.

[1] https://developers.Facebook.com/apps#_=_
[2] http://developers.Facebook.com/docs/authentication/permissions

8. L'API Graph Facebook répondra en fournissant une structure représentant les données de l'utilisateur sous une forme appropriée de façon à pouvoir facilement utiliser cette information. D'autres informations peuvent être obtenues dans un document disponible sur Facebook [Facebook, 2011c].
9. La dernière étape consiste à renvoyer le résultat de la page à l'utilisateur.

Le flux 2 permet l'authentification de l'utilisateur et la validation des autorisations de l'application par celui-ci. Le flux 7 permet de vérifier l'authenticité du site en comparant le paramètre *redirect_uri* et l'URL du site configurée dans le Developer App.

Le logout

Le logout consiste simplement à faire appel à la fonction FB.logout. Il s'agit au même titre que pour le côté serveur d'une déconnexion unique. Aucun accès ultérieur aux données n'est plus possible.

4.6 Réflexion sur les acteurs repris dans l'état de l'art

Facebook est donc un réseau social, et donc un fournisseur de services, prenant également le rôle d'un fournisseur d'identité par l'intermédiaire de son API Facebook Connect. De par son succès, sa popularité et son étendue, il est placé dans une position dominante en comparaison avec les autres fournisseurs d'identité. Cependant, comme nous pourrons le voir dans les sections suivantes, il perd de sa valeur de par les informations sensibles qu'il possède pour les personnes informées et soucieuses de leur vie privée. Malheureusement le panel de ces personnes concernées est relativement maigre. Une analyse de la confidentialité et de la vie privée de ce module, est étudiée dans les sections suivantes.

4.7 Considérations en termes de sécurité

Nous partons du postulat que le réseau interne utilisé et le poste client sont sécurisés et fiables. En effet, on constatera qu'une grande partie des attaques menées à l'encontre de l'utilisateur le sont par l'intermédiaire de l'ordinateur lui-même (hameçonnage, espions,…) ou du réseau (écoute du réseau, attaques de type MIM,…). Les plus grandes failles de sécurité sont liées à une mauvaise sécurité du côté de l'utilisateur.

4.7.1 Remarques communes aux deux flux

L'aspect sécurité étudié ici porte principalement sur les communications entre les 3 acteurs que sont l'utilisateur, Facebook, et le fournisseur de services utilisant l'API Facebook Connect. Voici quelques remarques communes aux deux types de flux:

- après différents tests effectués, on constate, conformément à la documentation reçue de Facebook, qu'aucun échange d'informations sensibles n'est effectué entre l'utilisateur et le fournisseur de services. La seule pouvant être remise en question est le *client_id* représentant l'identifiant unique de l'utilisateur Facebook. L'email échangé dans certains cas de figure, est, quant à lui, comparé sur base d'un hash :
    - Extrait de la documentation Facebook :
      « Le site web envoie une version cryptée de votre adresse électronique que nous mettons en correspondance dans une base de données d'adresses électroniques, que nous avons également cryptées. En cas de correspondance, nous transmettons au site web le nom d'utilisateur correspondant à l'adresse électronique. De cette manière, quand vous vous connectez au site à partir de Facebook, le site peut relier votre compte Facebook à votre compte sur ce site. » [Facebook, 2011d] *;*

- certaines publications [Lei et al, 2010] semblent pointer du doigt les modules d'authentification utilisant les adresses email comme identifiant unique comme un danger, pratique courante sur Facebook et d'autres sites.

4.7.2 Sécurité du flux coté serveur

Vue de l'extérieur, l'API Facebook Connect est sécurisée mais possède tout de même certaines lacunes. Notons les éléments suivants :

- aucun jeton de rafraichissement n'est fourni, le jeton d'accès n'a pas d'expiration (pas de paramètre *expires_in)* et est révoqué lors de la déconnexion de l'utilisateur ;
- le flux coté serveur représente le flux « code d'autorisation » de OAuth v2. Mais notre vue limitée sur le fonctionnement interne ne nous permet pas de dire s'il est en effet respecté comme il se doit et si toutes les précautions (authentification du client, utilisation d'un code d'autorisation et s'il existe, vérification de sa validité,…) ont bel et bien été prises.

4.7.3 Sécurité du flux coté client

L'authentification coté client existe en mode synchrone et asynchrone. Le mode asynchrone a été implémenté afin d'analyser l'aspect sécurité. En effet, le mode synchrone est utilisé pour debugger les applications mais résulte en de mauvaises performances et un impact non négligeable sur la navigation de l'utilisateur, raison pour laquelle ce type de flux ne sera que très rarement utilisé. Le choix du mode s'effectue à l'initialisation de l'API et peut donc être très facilement modifié.

Le flux « coté client », représentant le flux d'octroi implicite du protocole OAuth v2, est moins recommandé que le flux coté serveur (représentant le flux « code d'autorisation » dans le protocole OAuth v2). Les inconvénients sont mentionnés dans l'étude du protocole. Hormis ces aspects remarquons également que :

- le paramètre state tel que recommandé par OAuth 2.0, et contrairement au flux coté serveur, n'existe pas. L'implémentation s'expose donc à des attaques de type CSRF ;
- le paramètre *token_type* n'est pas présent dans la réponse du jeton d'accès ce qui n'a aucun impact à proprement parler si ce n'est une désinformation du client sur l'authentification supportée;
- une majeure partie de l'authentification est effectuée par l'intermédiaire du navigateur. La sécurité est donc principalement reportée soit sur l'utilisateur, soit sur Facebook lui-même ;

4.7.4 Sécurité de l'accès aux ressources

Contrairement aux recommandations de OAuth v2, le jeton d'accès peut être utilisé par n'importe qui tant que l'utilisateur est connecté. Il n'inclut, en effet, aucune information liée à la machine de l'utilisateur et ne semble pas effectuer d'identification du client. Si le réseau n'est pas sécurisé, une personne malintentionnée écoutant le réseau peut très rapidement et très facilement accéder aux ressources protégées de l'utilisateur s'il est toujours connecté.

4.7.5 Résistance aux attaques

Cette section a pour but d'établir une liste des principales attaques auxquelles Facebook Connect semble plus fragile. Ces attaques portent principalement sur l'aspect de login/logout et le jeton d'accès (aspects techniques). Les attaques, ayant pour source une négligence de l'utilisateur vis-à-vis de la sécurité du poste ou du réseau utilisé, n'ont pas été reprises (aspects humains, ex : spyware, hameçonnage,…). Toutes les recommandations à ce sujet peuvent être retrouvées dans la spécification de OAuth 2.0.

Attaque par rejeu

La sécurité du protocole réside dans l'importance que l'utilisateur et le fournisseur de services vont accorder au jeton d'accès et à sa conservation (non-distribution). Ce dernier ne contient aucune information (telles que IP, MAC address,…) sur la source qui peut avoir accès aux ressources. Dès lors quiconque possède le jeton d'accès pendant la session active de l'utilisateur peut avoir accès à ses informations et ce quelque soit le poste depuis lequel la requête est exécutée.

Il est d'autant plus important de conserver le secret du jeton d'accès si la variable « *offline_access* » est utilisée. Dans ce cas, une personne tierce pourrait très facilement avoir accès aux informations de l'utilisateur et ce, pendant un laps de temps illimité.

Divers

A noter qu'une authentification via Facebook ne permet pas d'accéder immédiatement à la partie Développeurs de Facebook. En effet, une confirmation de mot de passe vous sera demandée afin d'éviter tout accès frauduleux depuis une session active.

4.8 Respect de la vie privée

La vie privée est le cheval de bataille de bon nombre de spécialistes de la sécurité. Facebook n'échappe certainement pas à la règle et doit, au contraire, faire l'objet d'une attention particulière, raison de cette section. La partie authentification de Facebook Connect est relativement sûre à ce niveau. Cependant une fois celle-ci achevée, un ensemble de paramètres concernant les autorisations sont échangés.

La partie ci-dessous critique principalement l'atteinte à la vie privée. Cet aspect est primordial dans le cas d'un fournisseur d'identité agissant comme réseau social. Facebook possède, en effet, un grand nombre d'informations privées [Dratwa, 2010] et ce, souvent, de votre plus tendre enfance jusqu'à vos dernières activités, de vos relations familiales à vos relations professionnelles, sans oublier vos envies, préférences, loisirs,…

La documentation concernant ce à quoi l'utilisateur s'expose en donnant son autorisation (et même simplement en s'inscrivant sur Facebook), est suffisamment importante que pour décourager l'utilisateur à les lire. Les trois principaux liens de règlements et autres chartes de Facebook sont :

- les conditions d'utilisation[1];
- la charte de confidentialité[2] ;
- la politique à appliquer par les développeurs[3].

Les informations retenues par Facebook sont multiples et en y adhérant l'utilisateur lui donne implicitement le droit d'utiliser ses données bien qu'il en reste le propriétaire.

Facebook stipule explicitement que toutes informations rendues et définies comme publiques sont accessibles par « Tout le monde, y compris les personnes en dehors de Facebook » [Facebook, 2011e]. Les données concernées, dont la visibilité est inchangeable, sont votre nom, votre image de profil, vos réseaux, votre nom d'utilisateur et votre identifiant. De plus, toute information qui ne dispose pas d'une icône de partage doit être considérée comme publique [Zimmer, 2009]. Cela concerne, entre autres, tout ce qui est publié sur les murs et autres groupes.

Pour l'implémentation du flux coté serveur de Facebook, les développeurs reçoivent également les informations suivantes: votre tranche d'âge, vos paramètres régionaux, votre sexe. En plus de ces informations, ils ont accès à la liste de vos amis et donc à leur informations définies sous le panneau de configuration approprié [Facebook, 2011f]. Les informations suivantes sont transmises par défaut : statut courant, ville actuelle, formation et emploi, l'activité liée aux applications.

[1] https://www.Facebook.com/terms.php
[2] http://www.Facebook.com/privacy/explanation
[3] http://developers.Facebook.com/policy/

Vous trouverez à la Figure 4-6, un aperçu du panneau de configuration vous permettant de configurer les informations vous concernant que vos amis peuvent transmettre à des applications tierses. Par défaut et lors de la création d'un nouveau compte, la presque totalité des ces informations est cochée.

Figure 4-6 : Panneau de configuration Facebook des informations transmises par nos amis

Par ailleurs Facebook conserve également au-delà des informations rendues publiques les habitudes de navigation de l'utilisateur (afin de cibler plus précisément le contenu qui vous sera présenté), les publicités sur lesquelles vous cliquez, vos coordonnées GPS et bien d'autres informations [Facebook, 2011g].

Par défaut un compte supprimé est en fait désactivé et n'est pas réellement supprimé de Facebook. Il conserve donc l'intégralité de vos données pour une possible réactivation ultérieure. En cas de suppression, elle ne sera effective entièrement que 90 jours après l'introduction et la validation de cette demande.

Il semblerait que Facebook conserve d'autres données que celles mentionnées dans les chartes et autres règlements. Comme on peut le voir sur la Figure 4-7, il semblerait que nos anciens mots de passe (le hash du mot de passe) soient également conservés… Cette information semble avoir été conservée un mois environ.

Figure 4-7 : Tentative de connexion Facebook après modification du mot de passe

Facebook transmet les informations des utilisateurs à *« vous et à d'autres utilisateurs, tels que vos amis, les annonceurs qui achètent des publicités sur le site, et les développeurs qui conçoivent les jeux, les applications et les sites Web que vous utilisez.» [Facebook, 2011h].*

Au vu de la quantité d'informations détenues par Facebook, il est plus que primordial pour la sécurité et la vie privée des utilisateurs que ceux-ci prennent les dispositions nécessaires afin de ne fournir que le strict minimum des informations les concernant.

Un article récent nous rappelle qu'il faut rester vigilant et pour cause « les utilisateurs devraient faire attention à ce qu'ils écrivent sur leurs profils, car les juges peuvent utiliser ces éléments pour décider de la garde des enfants ou trancher lors de disputes financières » [Rob, 2012]. Les publications postées sur Facebook ne sont donc pas sans danger!

La meilleure solution reste de ne pas s'y inscrire au prix, malheureusement, de se voir exclure d'une communauté de plus en plus importante qui utilise Facebook comme média de communication entre les gens. Il faut choisir entre conserver sa vie privée et ne pas être informé concernant les derniers évènements sociaux, les invitations diverses,…

Gardons également à l'esprit que, par défaut, les nouveaux profils créés sont rendus publics. Ces informations publiques peuvent, en effet, servir à une personne malintentionnée pour tenter de pirater le compte d'un utilisateur par une méthode simple consistant à répondre aux questions de sécurité pour retrouver le mot de passe d'un utilisateur à l'aide des informations postées sur son profil pratiquant ainsi une usurpation d'identité.

Comme l'a dit, Jean-Philippe Moigny dans certains de ses nombreux travaux de recherche, il faut :

- « Toujours bien garder à l'esprit que ce qu'on publie devient public et que sur Internet, tout ce que vous ne contrôlez pas est définitivement hors de votre portée » [Moiny, 2011] ;
- « En droit, cliquer peut signifier conclure un contrat» [Moiny, 2009].

Pour information, 600000 tentatives de connexion quotidiennes à Facebook sont générées par des programmes automatiques ou par des outils de pirates qui ont pour seul objectif de violer la vie privée d'autrui [Raynal, 2011]. D'ou l'importance d'associer à son compte Facebook un mot de passe complexe comprenant des lettres, chiffres, majuscules, minuscules et caractères spéciaux.

De par les éléments repris ci-dessus, le développeur d'un fournisseur de services utilisant Facebook comme fournisseur d'identité et ayant implémenté le flux coté serveur de Facebook Connect est donc par excellence la personne détenant le plus facilement un grand nombre d'informations sur votre vie privée. Bien que celui-ci soit tenu de respecter le règlement de Facebook, il n'en reste pas moins un danger pour votre vie privée. Ces informations privées peuvent également être rendues accessibles même lorsque vous n'êtes plus connecté au site (via la variable scope ayant pour valeur « offline_access »*)*.

Toutes ces informations ne sont protégées que par des documents législatifs et peuvent aisément être conservées sur d'autres serveurs. Elles peuvent donc être rendues publiques de façon illégale. Les autorisations et la visibilité de vos informations sont la clé de votre vie privée au sein de Facebook.

N'oublions pas également que toute information rendue publique peut être, à tout instant, référencée par un moteur de recherche. Ces informations bien que rendues privées entretemps, resteront en libre accès sur la toile.

Rappelons finalement deux choses :

- Facebook présente un cas de *browsewrap agreement* [Moiny, 2010]. En d'autres termes, un contrat qui est conclu par la navigation même sur le site contrairement au *clickwrap* qui lui, consiste à cliquer explicitement sur un

bouton “J'accepte” avant de bénéficier des services fournis. Cet accord peut être considéré comme un contrat passé entre le fournisseur de services et le client ;
- le choix de vos paramètres de confidentialités est plus que déterminants en droit. Limiter au maximum les accès (et de préférence uniquement à ses amis).

Facebook est donc, en termes de vie privée, un vrai danger même si parfois cela n'apporte pas que des aspects négatifs. Récemment, Facebook a permis à la justice belge de retrouver deux braqueurs [Detout, 2012].

4.9 Critiques diverses

Administration de l'application

L'interface utilisateur pour l'administration du module est relativement bien faite. Après une lecture attentive de la documentation disponible sur le site, tout est relativement clair et facile à prendre en main.

Facilité de mise en œuvre

La plate-forme Facebook Developer est bien documentée. Il est très simple de s'y retrouver et de pouvoir parcourir les différents chapitres pour l'implémentation de Facebook Connect. De plus, Facebook met à disposition de l'utilisateur différents kits de développement[1], qui rendent l'utilisation de leur plate-forme relativement aisée.

Compatibilité des modules

Les aspects autorisation et authentification sont intégrés dans les SDK du réseau social. Facebook reste relativement fermé. Et la plate-forme pourrait presque être considérée comme une plate-forme propriétaire. Dans cette optique, il est difficile de rendre les différents modules de Facebook Connect compatibles avec d'autres fournisseurs d'identité, bien qu'ils soient basés sur un standard ouvert (OAuth).

Par ailleurs et pour information, Facebook est utilisé par d'autres fournisseurs d'identité pour s'authentifier : Yahoo, YouTube,…

4.10 Synthèse

Facebook est une plate-forme relativement fermée et proche d'une plate-forme « propriétaire ». Elle implémente, depuis peu, le protocole OAuth v2.

Cette plate-forme, bien que sécurisée et techniquement relativement sûre et fiable, pose néanmoins plusieurs problèmes liés à la vie privée. Ces problèmes sont pour la plupart du temps imputables à la négligence des utilisateurs et à des politiques de mise à disposition d'informations par défaut relativement larges. Tout ceci est décrit en détail dans de nombreuses chartes et autres règlements. Malheureusement le nombre important de ces documents peut rapidement en décourager la lecture par l'utilisateur. Facebook est ainsi couvert de tout type d'attaques juridiques lui permettant de faire de vos informations et votre vie privée son gagne-pain.

Le respect de ces règles est pourtant le b.a.-ba de la confidentialité sur Facebook. En effet, sans application de ces règles par tous, la confidentialité de nos données est mise à l'épreuve.

Côté respect du protocole OAuth v2, Facebook Connect respecte les principales recommandations mais possède néanmoins certaines failles et problèmes de sécurité, principalement liés à l'accès aux ressources avec le jeton.

Notons également que le flux implicite (flux « coté client » de Facebook) est nettement plus performant aux dépens, hélas, d'implications non négligeables d'un point de vue de la sécurité. Il est donc préférable d'utiliser le flux « côté serveur ».

[1] http://developers.facebook.com/docs/sdks/

5 OpenID 2.0

5.1 Introduction

OpenID est une norme, provenant du Web 2.0, très intéressante dans le cadre de services hébergés. La première version de ce protocole a été développée en mai 2005 et a été, ensuite, révisée afin d'aboutir à une version 2.0 finalisée en décembre 2007. Depuis cette date plus aucune mise à jour de ce protocole n'a été effectuée.

OpenID est un système d'authentification unique libre et décentralisé. Il permet, dans certaines implémentations, de créer différents profils chez le fournisseur d'identité permettant ainsi de sélectionner les informations que l'on souhaite divulguer au fournisseur de services. Pour ce faire, il requiert certaines relations de confiance préalablement établies avec le site implémentant ce système et les fournisseurs d'identité. On parle de modèle dit de confiance asymétrique : c'est le serveur d'identité en qui vous placez votre confiance qui s'occupe de vous authentifier. Il permet de transférer différents types d'attributs tout en utilisant un système d'authentification forte.

Les identifiants sont présentés au fournisseur de services sous forme d'un URI en lieu et place de fournir un couple identifiant/mot de passe.

Par rapport à son homologue en version 1[Giorgetti et Ducamp, 2008], OpenID 2.0 :

- est plus sécurisé ;
- est mieux documenté ;
- étend l'interopérabilité avec d'autres protocoles ;
- permet le partage d'attributs divers ;
- offre la possibilité de s'authentifier avec un nom de domaine ;
- est compatible XRI [Reed et McAlpin, 2005].

Différents exemples de fournisseurs d'identité sont repris à la Figure 11-7 en annexe de ce mémoire. Cette figure illustre un tableau comparatif de ces différents fournisseurs sur base de critères divers.

Un des premiers grands acteurs à avoir implémenté OpenID 2.0 en janvier 2008 est Yahoo[1]. Ensuite, petit à petit, cette norme s'est vue implémenter par plusieurs acteurs clés, dont Facebook[2], Dailymotion (maintenant à l'abandon au profit de Facebook Connect), Microsoft (via Geneva[3]) et bien d'autres encore.

Le manque de publicité, la méfiance des utilisateurs et son adoption « lente » par des acteurs clés n'ont pas facilité l'intégration d'OpenID 2.0. En conséquence, il est encore peu connu et peu reconnu. Il est parfois même abandonné par certains acteurs aux dépens d'autres modules d'authentification unique. Il est probable que le départ de Snorri Giorgetti, ex-président d'OpenID Europe, aurait favorisé son extinction dans une partie de l'Europe. D'autres parts, son manque d'implémentation par les fournisseurs d'identité (appelé fournisseurs OpenID) constitue un sérieux frein à son déploiement en masse.

A l'inverse, d'autres acteurs placent OpenID sur l'avant de la scène, comme en Estonie, où OpenID [Ideelabor, 2011] est devenu, grâce à la société IdeeLabor, le

[1] http://openid.yahoo.com/
Pour les développeurs : http://developer.yahoo.com/openid/
[2] https://www.facebook.com/settings?tab=account&section=linked&t
[3] http://connect.microsoft.com/site642

fournisseur d'identité du gouvernement. Il implémente une authentification via Eid ou encore via la carte SIM des mobiles.

5.2 Type et composition

Les acteurs impliqués dans le protocole OpenID 2.0 sont les suivants :

- **le consommateur** (**Relying Party**) : **fournisseur de services** désirant recevoir une preuve d'identité de l'utilisateur de la part du fournisseur d'identité ;
- le **fournisseur d'identité** (**OpenID Provider**) : **serveur d'authentification** OpenID ;
- **le client** : la plupart du temps il s'agit d'un navigateur web (**user-agent**).

Vous remarquerez que le client, dans cette spécification, représente **l'utilisateur** et non plus le fournisseur de services (quand le service est repris comme la partie « client » d'une communication client-serveur) comme c'était le cas dans les spécifications OAuth.

OpenID est constitué d'un composant élémentaire : OpenID Authentication (dans sa version 1.1 ou 2.0). Comme son nom l'indique, ce composant permet d'effectuer une authentification avec OpenID. A ce composant peut venir se greffer une série d'extensions, dont voici les principales:

- OpenID Provider Authentication Policy Extension 1.0 est une extension fournissant :
  - un mécanisme permettant de forcer l'utilisateur à utiliser un mode d'authentification précis ;
  - un mécanisme avertissant le fournisseur de services des polices de sécurité qui ont été utilisées pour l'authentification ;
  - un mécanisme permettant au fournisseur de services de demander le niveau d'authentification utilisé au fournisseur OpenID ;
- OpenID Attribute Exchange 1.0 : permet, comme son nom l'indique, d'échanger des informations d'identité entre deux endpoints (valable dans la version 2.0 uniquement). Les deux mécanismes proposés sont « fetch » (fournir des informations d'identité) et « store » (stocker des informations d'identité) ;
- OpenID Simple registration 1.0 : extension du protocole d'authentification permettant l'échange de données de profil (8 informations peuvent être transmises : le pseudo, l'email, le nom, la date d'anniversaire, le genre, le code postal, le pays, la langue, le fuseau horaire).

Les spécifications de ces différents composants peuvent être retrouvées sur le site officiel[1] d'OpenID. Nous ne nous attarderons pas sur ces différentes extensions facultatives. Bien que celles-ci soient très intéressantes et permettent une multitude de fonctionnalités complémentaires, elles ne seront pas vues en détail. Seule celle concernant l'authentification [Ferg et al, 2007] nous intéresse dans le cadre de ce mémoire.

OpenID permet donc de :

- remplir des formulaires d'enregistrement sur les sites web supportant OpenID avec les informations du profil que le fournisseur d'identité détient ;
- utiliser le compte unique d'un fournisseur d'identité pour s'authentifier dans plusieurs milliers d'applications ;
- gérer plusieurs identités ;
- contrôler les informations divulguées par le fournisseur d'identité ;
- minimiser les risques de sécurité liés au mot de passe (changer son « portefeuille de mot de passe » en une seule fois, …) ;

[1] http://openid.net/developers/specs

- ne retenir qu'une combinaison utilisateur/ mot de passe.

Certaines de ces fonctionnalités ne sont pas d'office applicables à tous les fournisseurs d'identité. Ceci dépend de la mise en place, par ces derniers, des différentes extensions citées précédemment.

5.3 Concepts de base

Vous trouverez un schéma du flux abstrait du protocole OpenID à la Figure 5-1 et les détails concernant chacune des étapes en dessous de celui-ci. Pour rappel et comme mentionné dans l'état de l'art, l'OP est l'OpenID Provider et donc le fournisseur d'identité alors que le RP est le Relying Party représentant le fournisseur de services.

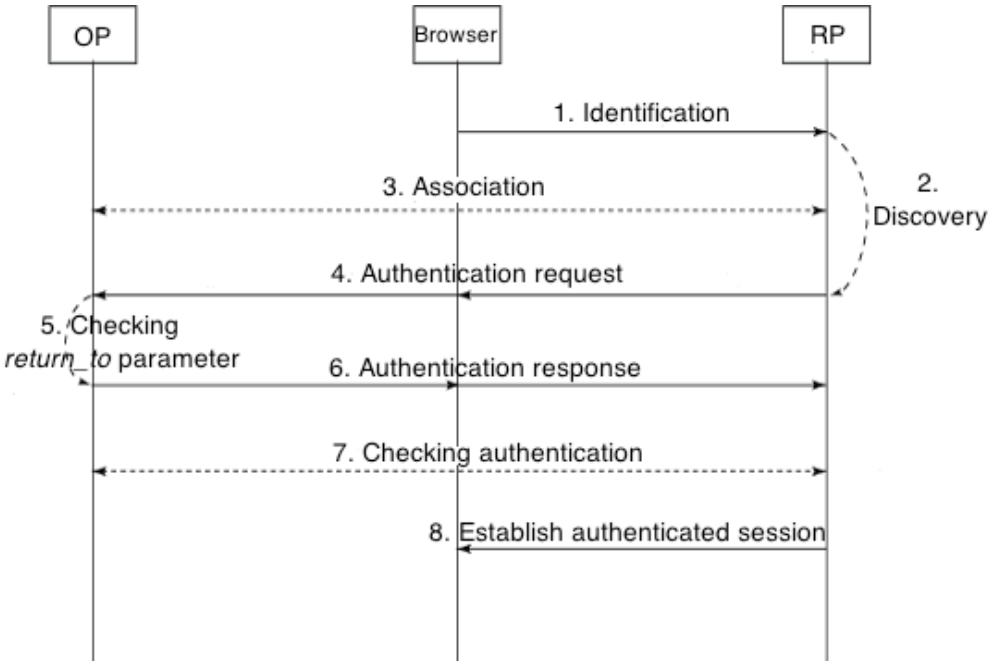


Figure 5-1 : Flux abstrait du protocole OpenID 2.0 [Walther, 2011]

1. L'utilisateur initie l'authentification en présentant ses identifiants OpenID (identifiants représentés sous forme d'un URI, par exemple : *x.mydomain.be*) au fournisseur de services (Relying Party).
2. Après avoir normalisé les identifiants de l'utilisateur, le fournisseur de services effectue une découverte de l'utilisateur et établit l'URL du endpoint du fournisseur OpenID que l'utilisateur devra utiliser (les identifiants de l'utilisateur peuvent, évidemment, provenir du fournisseur OpenID).
3. (facultatif) Le fournisseur de services et le fournisseur OpenID établissent un secret partagé (association). Le fournisseur OpenID l'utilisera pour signer les différents messages. Le fournisseur de services utilisera ce secret pour les vérifier.
4. Le fournisseur de services redirige le navigateur vers le fournisseur OpenID avec une requête d'authentification OpenID.
5. Le fournisseur OpenID établit la liaison si :
    - l'utilisateur l'autorise à effectuer l'authentification OpenID ;
    - l'utilisateur en a le souhait (c'est également à cette étape que l'utilisateur donne l'autorisation d'accéder à certaines de ses données).
6. Le fournisseur OpenID redirige le navigateur vers le fournisseur de services avec une assertion attestant que l'authentification est valide ou un message comme quoi elle a échoué.

7. Le fournisseur de services vérifie l'information reçue du fournisseur OpenID (vérification de l'URL de retour, de l'information découverte, du nonce et de la signature) en utilisant la clé partagée établie durant la phase « association » ou en envoyant directement une requête au fournisseur OpenID.
8. Le fournisseur de services établit une session authentifiée avec l'utilisateur.

5.4 Etude du protocole

L'étude portera principalement sur le module OpenID Authentication v2.0 datant du 5 décembre 2007.

5.4.1 Flux de communication

Il existe deux types de communication au sein d'OpenID Authentication 2.0 :

- **communication directe :** type de communication initié par le fournisseur de services en direction du fournisseur d'identité. Il est utilisé pour :
  - établir des secrets partagés ;
  - vérifier les assertions d'authentification.

- **communication indirecte :** type de communication initié soit par le fournisseur de services, soit par le fournisseur d'identité. Il est utilisé pour effectuer:
  - des requêtes d'authentification ;
  - des réponses d'authentification.

  Il existe deux méthodes de communication indirecte :
  - redirections HTTP : la redirection permet de transférer des données par l'intermédiaire de redirections et contient le message d'authentification OpenID ;
  - soumission d'un formulaire HTML : un ensemble de clés-valeurs peuvent être transmises par l'intermédiaire d'un formulaire HTML.

Ces deux modes nécessitent une connaissance de l'URL du fournisseur de services ou du fournisseur d'identité et qu'ils soient prêts à recevoir ce type de message. La personne étant l'initiatrice de la communication choisira la méthode de communication suivant ses capacités, la taille du message ou tout autre facteur externe.

5.4.2 Flux d'authentification

Le flux repris à la Figure 5-2 représente le flux d'authentification OpenID 2.0. Les différentes étapes de ce flux sont détaillées en dessous de cette figure.

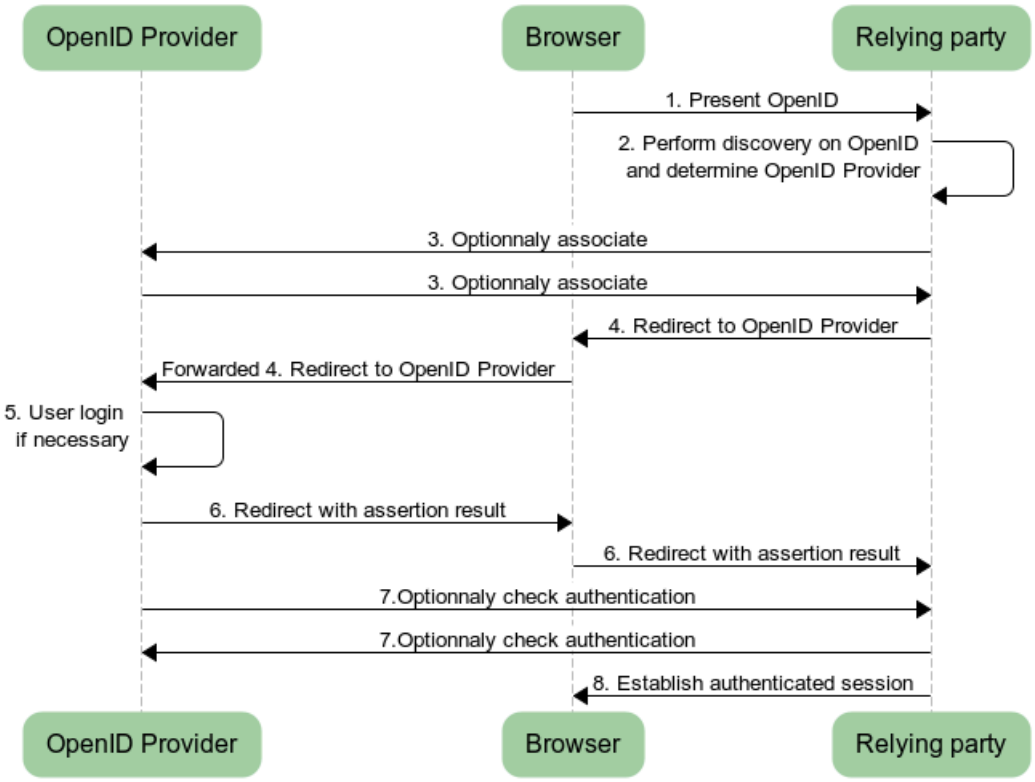


Figure 5-2 : Flux d'authentification OpenID 2.0

1. Afin de pouvoir offrir un support pour l'authentification, le serveur OpenID doit proposer un formulaire d'authentification incluant un champ d'insertion de l'identifiant de l'utilisateur. L'identifiant introduit doit être normalisé comme décrit dans la spécification d'OpenID 2.0 [Ferg et al, 2007].
2. Avant de décrire le processus de découverte, il nous faut introduire la notion de document XRDS. Il s'agit d'un document XML reprenant des entrées pour les services liés à l'identification. Il contient :
    - l'identifiant du fournisseur OpenID : recherché en premier par le fournisseur de services ;
    - un identifiant clamé.

    Le processus de découverte consiste, pour un fournisseur de services, à utiliser l'identifiant, réceptionné à la première étape, afin de découvrir les informations nécessaires à l'initialisation des requêtes. Ce processus s'effectue comme suit.
    a. Si l'identifiant est un XRI [Reed et McAlpin, 2005], la résolution XRI fournira un document XRDS contenant toutes les informations nécessaires. L'élément contenant l'authentification OpenID doit également contenir un identifiant clamé. Il s'agit d'un point vital en termes de sécurité car un des principaux objectifs de cet élément est de faire valoir un identifiant persistant qui ne sera jamais réassigné, en s'assurant donc que le XRI n'est pas repris par un nouvel utilisateur. Remarque : certains fournisseurs de services peuvent tirer un avantage à utiliser des proxys de résolution XRI car ils permettent de se passer du fournisseur de services pour effectuer une résolution XRI localement.
    b. Si l'identifiant est une URL, le protocole Yadis [Miller, 2006] devrait tenter de retirer le document XRDS. Lorsque le document a pu être récupéré, les éléments sont traités de façon similaire à un XRI.
    c. Si le protocole Yadis échoue et qu'aucun document XRDS valide n'a pu être obtenu, une tentative de découverte basée sur le protocole HTML devrait être effectuée. Ce type de découverte doit être supporté par le

fournisseur de services (via un document HTML disponible à l'URL appropriée) et est uniquement utilisé pour découvrir les identifiants clamés. Le serveur hébergeant le document HTML peut être différent du serveur de fournisseur OpenID de l'utilisateur.

L'URL du endpoint du fournisseur d'identité OpenID ainsi que la version du protocole seront récupérées si le processus s'achève correctement. L'identifiant clamé et l'identifiant local du fournisseur OpenID seront également présents si l'utilisateur n'a introduit aucun identifiant.

3. Une association entre un fournisseur de services et un fournisseur OpenID est un secret partagé entre ces deux acteurs, utilisé pour vérifier les messages, minimisant ainsi les aller-retours (recommandé mais non obligatoire).
   Une session d'association est initiée par une communication directe depuis le fournisseur de services vers l'URL du endpoint du fournisseur OpenID. En réponse, le fournisseur de services renverra un formulaire (de couples clé-valeur) ou une erreur si le fournisseur OpenID ne supporte pas un type de session ou un type d'association particulier. Si aucune association n'a été établie, le fournisseur de services peut continuer le processus d'authentification à l'aide de la communication directe.
   Remarque : les associations permettent d'éliminer le besoin de communication directe afin de vérifier la signature après chaque requête/réponse d'authentification.
4. L'étape suivante consiste à envoyer une requête d'authentification vers le fournisseur OpenID afin d'obtenir une assertion. Ce type de requête est une communication de type indirecte.
5. L'utilisateur s'authentifie si nécessaire. Les fournisseurs OpenID devraient, quant à eux, vérifier que l'URL spécifiée au paramètre *return_to* de la requête est bien l'URL du endpoint d'un fournisseur de services en effectuant une découverte sur ce dernier. Si les deux URL sont équivalentes, la vérification est réussie autrement elle échoue.
6. Si un utilisateur s'authentifie avec succès, le fournisseur OpenID devrait envoyer une assertion positive au fournisseur de services en utilisant une redirection autrement il renverra une assertion négative si le fournisseur OpenID n'est pas capable d'identifier l'utilisateur ou si celui-ci n'approuve pas la requête d'authentification.
   Si le fournisseur de services a demandé un identifiant dirigé par le fournisseur OpenID et qu'il existe différents identifiants pour lesquels l'utilisateur est autorisé à fournir des réponses d'authentification, le fournisseur OpenID devrait permettre à l'utilisateur de choisir l'identifiant à utiliser.
   Si la valeur du paramètre *openid.return_to* n'est pas présente dans la requête, le fournisseur de services ne souhaite pas recevoir une assertion d'authentification du fournisseur OpenID. Cela peut être utile lorsque l'on utilise des extensions pour transférer les données depuis le fournisseur de services jusqu'au fournisseur OpenID.
7. Lorsqu'un fournisseur de services reçoit une assertion positive, il doit vérifier les points suivants avant d'accepter l'assertion :
   - la valeur de l'URL comprise dans le paramètre *openid.return_to* correspond à la requête courante ;
   - l'information découverte correspond à l'information présente dans l'assertion ;
   - une assertion n'a pas encore été acceptée de ce fournisseur d'identité ;
   - la signature de l'assertion est valide et tous les champs requérant une signature ont été signés.

   L'authentification sera effective lorsque les 4 conditions mentionnées auront été remplies.

Remarque : pour éviter les attaques par rejeu, le fournisseur OpenID ne doit pas fournir plus d'une réponse par requête d'authentification (le nonce permettra de faire la correspondance entre la requête d'authentification et la réponse).

8. Etablissement de la session authentifiée avec le navigateur de l'utilisateur. L'identifiant clamé dans la réponse d'authentification devrait être stocké localement par le fournisseur de services en tant qu'information à propos d'un utilisateur. L'identifiant clamé devrait être utilisé comme un identifiant visible par l'utilisateur. Les fournisseurs OpenID avec un grand nombre d'utilisateurs peuvent recycler les identifiants s'ils le souhaitent. Il est recommandé qu'une redirection s'opère depuis l'URL HTTP vers l'URL HTTPS (pour des raisons de sécurité évidentes).

5.4.3 Découverte des tiers OpenID

La découverte des fournisseurs OpenID permet aux fournisseurs de services de découvrir les sites qui supportent OpenID. Elle permet aussi aux fournisseurs OpenID de vérifier automatiquement que l'URL du paramètre *return_to* présent dans une requête OpenID est identique au endpoint d'un partenaire de confiance OpenID.

Les fournisseurs de services devraient utiliser le protocole Yadis afin de publier leurs URLs *return_to* valides. Ainsi, les fournisseurs OpenID peuvent vérifier les URLs *return_to*.

Les points saillants du processus de découverte se trouvent à l'étape 2 de la section précédente.

5.4.4 Extensions du protocole OpenID v2

Une extension OpenID est un protocole qui se greffe sur la requête et la réponse d'authentification. Les extensions sont aussi bien utilisées pour fournir de l'information supplémentaire au sujet d'une requête, d'une réponse que sur le sujet de la réponse d'authentification.

Les extensions OpenID sont identifiées par un type d'URI ou un élément présent dans le document XRDS associé avec l'identifiant clamé. Le type d'URI est également utilisé pour associer les couples clé-valeur dans les messages utilisant les extensions. Pour ce faire, la clé doit être associée avec le type de l'URI. Une fois cette étape terminée, tous les couples de clés présents dans les messages sont associés avec cette extension.

Un namespace ne doit pas se voir assigné plus d'un alias dans le même message. Si un message est une réponse à un autre message, la réponse peut utiliser un alias différent pour faire référence au même namespace.

Les extensions ne doivent pas définir plusieurs paramètres avec le même nom. Celles qui nécessitent l'envoi de plusieurs valeurs dans le même paramètre doivent définir leur propre convention pour le faire.

Un exemple d'une telle extension est celle de MediaWiki[1].

5.5 Considérations en termes de sécurité

5.5.1 Prévention des attaques

Ecoute du réseau

Un pirate pourrait intercepter une assertion d'authentification valide et la réutiliser. Cette attaque peut être évitée :

- en utilisant une couche de transport chiffrée ;
- en vérifiant le *nonce* lorsqu'on effectue la vérification du message.

[1] http://www.mediawiki.org/wiki/Extension:OpenID

Les attaques de l'homme du milieu

Les associations permettent de se prévenir de la falsification des champs signés par un intrus placé au milieu d'une communication (excepté durant la phase de découverte, les sessions d'associations et la communication directe). L'altération de champs signés sans la clé partagée nécessite de violer la MAC [Hammer, 2012]. Actuellement, aucune attaque de ce type n'est connue dans ce protocole. La qualité de la protection dépend de l'aspect aléatoire de la clé MAC [Hammer, 2012] partagée.

Si la résolution DNS ou la couche de transport est compromise, la signature des messages n'est plus adéquate, puisque l'attaquant peut usurper l'identité du fournisseur OpenID et fournir ses propres associations, ou ses propres décisions.

Si un attaquant peut altérer le processus de découverte en altérant le document XRDS, il n'est plus nécessaire de placer quelqu'un au milieu de la communication. Pour se prévenir contre ce type d'attaque, il suffit de signer numériquement le fichier XRDS via XMLDSIG [Eastlake et al, 2002].

L'utilisation de SSL avec des certificats signés par une autorité de confiance permet de se prémunir contre les attaques de l'homme du milieu en vérifiant le résultat d'une recherche DNS sur le certificat. Une fois la validité du certificat établie, l'altération n'est plus possible. L'usurpation d'identité d'un serveur SSL nécessite une falsification ou le vol d'un certificat, ce qui est significativement plus difficile que les autres types d'attaque.

Afin d'obtenir une protection correcte, même si le protocole OpenID v2 ne l'exige pas, SSL est fortement recommandé pour toutes les transactions sans exception. Les bonnes pratiques courantes exigent qu'un fournisseur OpenID utilise SSL avec un certificat signé par une autorité de confiance. Suivants ses propres recommandations de sécurité, un fournisseur de services peut choisir de ne pas compléter, ou même de ne pas commencer, une transaction si SSL n'est pas utilisé correctement sur ses différents endpoints.

L'attaque de type MitM peut être effectuée sur un fournisseur de services compromis. Dans ce cas, il effectuera une découverte des identifiants clamés de l'utilisateur et, à la place de rediriger le user-agent vers le fournisseur OpenID, il fera passer l'ensemble des requêtes par un proxy dont il est le propriétaire. Afin de se prévenir de ce type d'attaque, il est indispensable que le fournisseur OpenID établisse un canal sécurisé avec l'utilisateur.

Les identifiants par URL HTTP et HTTPS

Il est primordial qu'un utilisateur arrivant sur un endpoint en HTTP soit automatiquement redirigé vers un endpoint sécurisé en HTTPS pour des raisons de sécurité évidentes. Les utilisateurs devraient quant à eux, introduire immédiatement l'URL en HTTPS sur chaque fournisseur de services afin d'éviter une redirection de l'utilisateur vers un endpoint compromis et donc plus précisément vers un fournisseur OpenID malintentionné.

Les attaques par déni de service

Au sein du protocole OpenID, il y a plusieurs entrées susceptibles d'être l'objet d'une attaque par déni de service en exemple une demande d'association, d'authentification ou de vérification d'une signature. L'une de ces phases potentiellement dangereuses est celle concernant l'association. En effet, durant cette phase le fournisseur OpenID doit exécuter une opération exponentielle. Un pirate peut, par l'intermédiaire de certaines techniques, forcer le fournisseur OpenID à effectuer cette opération en temps réel et de façon prioritaire pour chaque message avec les performances que cela implique.

Pour se défendre contre ce type d'attaque, les fournisseurs OpenID peuvent utiliser les adresses IP en limitant le quota de requêtes. Il existe également d'autres techniques se focalisant sur l'interdiction des requêtes.

Les variables de protocole

Les variables suivantes peuvent jouer un rôle sur la sécurité de l'utilisation du protocole :

- la présence de wildcards au sein des attributs ;
- la nécessité des associations prioritaires avant la requête d'authentification ;
- l'acceptation des types d'identifiants clamés ;
- l'autorisation des certificats « self-issued » pour l'authentification ;
- la signature du document XRDS ;
- le retrait d'un document XRDS via un canal sécurisé ;
- l'utilisation de type de session pour créer les associations ;
- la détention, par le fournisseur de services, d'un document XRDS ;
- l'acceptation de types d'associations par le fournisseur OpenID pour les signatures ;
- l'utilisation d'un canal sécurisé pour la requête d'association.

Hameçonnage

Les techniques d'anti-hameçonnage classiques ou l'utilisation de l'authentification à clés cryptographiques asymétriques permettent de se protéger contre l'hameçonnage [Bortzmeyer, 2007].

Autrement, seule une vérification visuelle de l'utilisation du protocole SSL et de l'URL permettra d'éviter les désagréments que le hameçonnage pourrait occasionner.

Le fournisseur de services devrait rediriger l'utilisateur vers l'URL de endpoint du fournisseur OpenID dans une fenêtre de navigateur au plus haut niveau avec tous les contrôles visibles. Cela permet une meilleure protection contre les attaques de hameçonnage.

Les fournisseurs OpenID devraient informer leurs utilisateurs de ce type d'attaque et devraient les équiper d'outils contre celle-ci (ex : plugins du navigateur permettant de vérifier l'authenticité de l'URL de endpoint du service d'authentification des fournisseurs OpenID).

User-agents

Aujourd'hui, plusieurs applications web et protocoles ont une entière confiance dans la sécurité du navigateur et les serveurs dont ils font usage. Malheureusement les users-agents peuvent être infectés de tous types de malware limitant ainsi la force des assertions d'authentification. En effet un malware peut très bien avoir validé pour l'utilisateur une requête précise,…

Afin de limiter un maximum ce type d'attaque et pour une meilleure sécurité, les fournisseurs OpenID devraient pouvoir fonctionner sans intégrer un quelconque script. Ceci permettrait d'éviter les attaques de types Cross-site-scripting.

CSRF

La défense contre ce type d'attaque doit également être implémentée sur le site auquel on se connecte. Ce problème n'est pas spécifique à OpenID mais il faut être d'autant plus prudent comme OpenID permet de faire de l'authentification unique et qu'il y a une étape primordiale d'authentification chez le fournisseur d'identité.

L'idéal étant évidemment de s'authentifier sur son fournisseur avant même d'attaquer le site cible, ainsi les risques liés à ce type d'attaque sont moindres.

Il est également possible de demander à certains fournisseurs d'identité de demander une confirmation d'identification afin de limiter les dangers liés à l'authentification automatique.

5.5.2 Failles de sécurité

En dehors du protocole lui-même, le problème (et l'avantage) fondamental d'OpenID est simple : tout dépend d'un simple couple d'identifiants [Kiani, 2011], et donc, principalement, de la sécurité de ceux-ci. Les failles de sécurité peuvent, entre autres, être liées à l'exploitation des défauts dans le processus de réinitialisation/recouvrement du mot de passe, afin de s'en servir pour usurper l'identité de la victime. L'étendue des éventuels dégâts en cas de réussite de l'attaque est donc très importante. La perte de ces identifiants est tout aussi importante.

Il existe, cependant, des contre-mesures telles que :

- l'utilisation d'un secret visuel permettant d'identifier le fournisseur ;
- les applications de type One Type Password[Kiani, 2011] (couteux en temps utilisateur mais efficace).

Comme déjà mentionné précédemment, les fonctionnalités qu'offrent OpenID le rendent plus sujet à des attaques par hameçonnage [The identity corner, 2007] : l'utilisateur devra donc rester attentif et vigilant lors des différentes redirections effectuées tout au long du processus.

Aucune faille de sécurité n'a, cependant et à ce jour, été découverte pour ce protocole. Cependant, selon un article [Journal du net, 2011] de la fondation OpenID, un des modules cités dans ce mémoire et décrit dans l'introduction possède une faille de sécurité. Il s'agit du module « Attribute Exchange », pour lequel la recommandation suivante a été prononcée par le porte-parole de la fondation: « Pour les applications vulnérables, nous recommandons en premier lieu de modifier leur code pour accepter seulement les valeurs d'attributs signées » [Journal du net, 2011]. Cette faille n'a semble-t-il pas encore été exploitée.

Une recherche [Laurie et Clayton, 2008] effectuée par les laboratoires de l'université de Cambridge pointe du doigt les systèmes d'authentification unique utilisant le service DNS. Il ajoute une couche supplémentaire de risque. Ils pointent également du doigt la possibilité d'utiliser des clés privées insuffisamment sécurisées au sein du protocole OpenID rendant les fournisseurs d'identité peu fiables. L'exploitation de ces deux dernières failles de sécurité aurait permis à une personne malintentionnée de récupérer la clé privée, et ainsi de se faire passer pour un fournisseur OpenID redirigeant, par l'intermédiaire du DNS, l'utilisateur vers le site compromis. Pour parvenir à exploiter une telle faille, une négligence évidente de la part du fournisseur OpenID et de l'utilisateur doit être d'application. La recherche précise évidemment qu'il est possible d'implémenter OpenID de façon beaucoup plus sécurisée que ce que n'exige le protocole comme pour la plupart de tous protocoles d'authentification unique.

La norme ne parle pas du Single Log Out, il est donc intéressant d'étudier le comportement de ce mécanisme dans les implémentations d'OpenID v2. L'avantage certain d'OpenID consiste également à désactiver l'ensemble des sessions authentifiées en une seule fois. Ceci dépend cependant de l'implémentation effectuée par le tiers. En effet, si celui-ci décide de garder une session ouverte sur base d'un id de session sans réinterroger le fournisseur OpenID, l'utilisateur pourrait toujours être authentifié à un service alors qu'il ne l'est plus chez le fournisseur d'identité… Cela peut ouvrir de dangereuses opportunités d'attaque par rejeu.

5.6 Respect de la vie privée

Au vu de la mission d'OpenID 2.0 (permettre l'authentification sur un fournisseur de services sans nécessité préalable d'établir un lien quelconque), de ses différentes

fonctionnalités et de ses avantages vis-à-vis de ses concurrents, il est difficile, pour ce protocole, de faire mieux en termes de respect de la vie privée [Prodromou, 2008].

Il faut cependant le manipuler avec précaution au vu des dangers auxquels il est plus facilement exposé (principalement le hameçonnage).

Les données personnelles sont de la responsabilité du fournisseur OpenID. Il est donc indispensable d'avoir une confiance fondée sur ce fournisseur et par précaution de lui fournir le minimum d'information nécessaire. Afin de garder la possibilité de changer facilement de fournisseur OpenID et pour tenter de garder, tant que possible, l'anonymat de son fournisseur d'identité, il peut être intéressant de mettre en place une page sur un serveur web dédié qui effectuerait une redirection vers le fournisseur OpenID. Pour rappel, OpenID 2.0 est un standard ouvert, il est donc possible que l'utilisateur soit son propre fournisseur d'identité.

Les données personnelles ne sont pas les seules informations sensibles dont dispose le fournisseur OpenID. Il a également en sa possession les habitudes de navigation de l'utilisateur. Ces informations pourraient être transmises à des fins de marketing ou autres. Il est donc important lors du choix du fournisseur OpenID de prendre ses points d'attention en considérations et d'envisager, dans certains cas, les possibilités d'implémenter son propre fournisseur OpenID.

En Estonie, un fournisseur OpenID [Ideelabor, 2011] est même utilisé pour fournir accès à des services aux utilisateurs via leur Eid permettant ainsi de vérifier les informations d'identification via l'infrastructure à clés publiques nationale. Ce mécanisme a été possible grâce à son intégration au sein de l'Eid Installer (application similaire à celle présente sur le portail eid.belgium.be en Belgique). Il est présent sur 40% des installations. Ce qui porte à croire que son utilisation peut-être sécurisée et respectueuse de la vie privée sous certaines conditions. Il offre, également, la possibilité à l'utilisateur de créer un compte unique anonyme en plus de son compte réel. Il est facile d'imaginer les informations sensibles qui peuvent être présentes sur une carte d'identité et le danger élevé lié à son utilisation. Tout porte donc à croire qu'OpenID est un protocole fiable.

Gardons en tête qu'avoir une identité OpenID différente ne signifie pas l'anonymat complet. En effet, des informations telles que l'adresse e-mail et la date de naissance peuvent vous identifier de manière unique. Mais un des points forts d'OpenID, rappelons-le, est également la gestion de multiples identités sous le même compte, il serait dommage de ne pas en faire usage quand cela est possible.

5.7 Synthèse

Plusieurs initiatives ont été mises en place afin qu'OpenID puisse se faire connaître du grand public mais sans vraiment percer de manière significative. Il progresse petit à petit et est, de plus en plus, implémenté par de grands acteurs. Il semble cependant devoir faire face à des acteurs qui semblent l'abandonner.

OpenID 2.0 peine à se faire adopter par certains fournisseurs de services et pour cause :

- les utilisateurs se méfient de devoir fournir tous leur identifiants à un fournisseur d'identité ;
- difficulté pour les sites tiers de s'assurer que le serveur d'identité n'a pas été compromis ;
- OpenID n'est pas encore suffisamment implémenté ;
- difficile de dire aux premiers abords si un site est éthiquement correct ;
- chaque donnée transférée à ce site est susceptible d'être divulguée sur le net ;
- l'aspect « anonymat » du protocole ne permet pas d'obtenir suffisamment d'informations probantes au sujet des utilisateurs, certains fournisseurs de services ne l'envisagent pas pour cette raison ;
- un identifiant perdu l'est pour des milliers d'applications.

La version2 du protocole offre une certaine souplesse quant aux possibilités d'authentification. Les avantages de cette solution sont non négligeables :

- simplicité d'utilisation ;
- complétion de formulaire d'enregistrement de façon très aisée ;
- possibilité de définir différents profils ;
- ….

L'apparition de multiples services d'authentification unique et de concurrents tels que BrowserID [Moonz, 2011] ne facilitera pas la percée d'OpenID. Ses lentes évolution et insertion dans les marchés risquent d'être un frein sérieux à son adoption. Difficile donc de prévoir quel sera son avenir d'autant plus incertain avec l'avenue d' OpenID Connect (décrit au chapitre suivant).

Le concept d'OpenID 2.0 est pourtant innovateur et si les liens de confiance sont forts, il permettrait d'ouvrir une communauté suffisamment importante que pour faire de l'authentification unique de manière large tout en laissant la liberté à l'utilisateur et aux tiers des fournisseurs d'identité de choisir en qui placer sa confiance. Le succès d'OpenID réside dans son adoption par de gros fournisseurs d'identité.

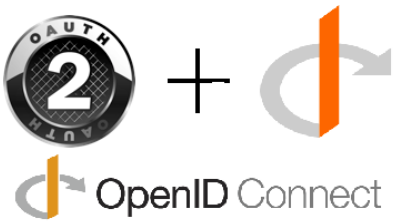
Il y a d'importantes initiatives qui ont vu le jour. L'implémentation d'OpenID par un fournisseur d'identité intégrant l'Eid (impliquant donc une collaboration avec l'état) en Estonie en est un bon exemple.

Avec OpenID, tout le monde peut devenir un fournisseur d'identité. C'est l'avantage d'un protocole ouvert avec, comme principal inconvénient, la fuite possible des informations enregistrées auprès de celui-ci ainsi que des habitudes de navigation (souvent à des fins de marketing…). Il est indispensable de partager le minimum d'informations nécessaires et ce même avec un fournisseur OpenID de confiance pour garantir la pérennité de sa vie privée. Le choix du fournisseur OpenID est donc primordial.

OpenID est un protocole très intéressant et apportant plusieurs avantages vis-à-vis des systèmes de fédérations concurrents. Il est cependant indispensable de le manipuler avec précaution et en connaissance de cause. Il est également primordial de se renseigner sur les fournisseurs de services qui l'utilisent et de ne l'utiliser que sur des implémentations suffisamment sécurisées. Il n'est donc pas à la portée de n'importe quel individu…

6 OpenID Connect 1.0

6.1 Introduction



Suivant un communiqué de la communauté OpenID [Sakimura, 2012], les différents membres ont approuvé à la presqu'unanimité, ce 7 février 2012, les différentes spécifications définissant OpenID Connect 1.0… Ces spécifications feront donc l'objet d'une dernière révision finale et d'un dernier vote, dans un avenir proche, avant de passer au stade de standard OIDF.

OpenID Connect est un protocole ayant été développé au-dessus de la pile OAuth 2.0 et incluant un service de découverte comme le fait OpenID 2.0. Ce protocole a été conçu pour répondre aux lenteurs d'OpenID (via le support JavaScript et les applications « rich-client ») et permettre d'inclure des mécanismes de chiffrement et de signature plus robustes tout en offrant un panel de possibilités d'extensions plus larges.

A l'heure d'écrire ces lignes, la communauté a achevé la rédaction des différents brouillons à destination des personnes souhaitant implémenter en test cette spécification et attend un feedback de leur part. L'étude de ce protocole portera donc sur les principales spécifications d'OpenID Connect.

OpenID Connect 1.0 est une tentative d'assemblage des meilleures pièces [Calore, 2010] d'OAuth v2 et d'OpenID 2.0 afin de créer un nouveau protocole dont le seul objectif est de simplifier son utilisation.

OpenID Connect 1.0 utilise donc:

- OpenID pour permettre à un utilisateur de prouver son identité ;
- OAuth pour permettre de fournir un accès à des ressources de l'utilisateur par l'intermédiaire d'API.

Quelques points qu'OpenID Connect cherche à améliorer :

- l'utilisation d'un URI comme identifiant au sein d'OpenID était source de confusions pour l'utilisateur. Afin de faciliter la prise en main par l'utilisateur lambda, OpenID Connect offre une prise en charge de l'utilisation de l'adresse email comme identifiant ;
- l'aspect anonymat qu'un utilisateur pouvait obtenir d'OpenID n'était pas vraiment intéressant pour les acteurs souhaitant collecter des informations sur leur client. OpenID Connect permettra de fournir de plus amples informations au fournisseur de services tout en laissant le contrôle de la divulgation de ses données à l'utilisateur ;
- la plupart des applications telles que Facebook, Twitter utilisent OAuth pour accéder aux ressources de l'utilisateur pour la raison citée au point précédent. Cependant, au vu de la complexité d'intégrer OpenID à OAuth 2.0, il était très difficile pour ces fournisseurs de services d'en offrir le support. OpenID Connect permettra de faciliter ces aspects.

6.2 Type et composition

OpenID Connect est donc une spécification modulaire composée d'un ensemble de spécifications plus légères fournissant un cadre d'interactions d'identités via des API's RESTful. Les développeurs peuvent donc effectuer la sélection des modules dont ils ont besoin pour leur implémentation.

Le déploiement le plus simple d'OpenID Connect permet à tout type de clients (navigateurs, JavaScript), de demander et de recevoir des informations sur les identités et les sessions actuellement authentifiées.

La suite de spécifications OpenID Connect :

- est extensible ;
- permet aux participants de supporter d'éventuels chiffrements de données d'identité ;
- d'effectuer une découverte de fournisseurs OpenID ;
- permet une gestion avancée des sessions.

OpenID Connect est très similaire à OpenID v2.0 en termes de fonctionnalités mais le fait d'une façon plus « API friendly ».

OpenID Connect 1.0 est une simple couche d'identité au dessus du protocole OAuth 2.0 permettant aux clients:

- de vérifier l'identité de l'utilisateur basée sur l'authentification effectuée par un serveur d'autorisation ;
- d'obtenir des informations basiques sur le profil de l'utilisateur d'une façon interopérable et suivant les règles RESTful.

OpenID Connect est composé de différentes spécifications à sélectionner suivant les fonctionnalités et l'objectif que l'on recherche. La Figure 6-1 résume bien cet aspect composite. Il représente un schéma d'imbrication des différentes spécifications d'OpenID Connect. OAuth 2.0 est repris en tant que couche inférieure à OpenID Connect. Ceci signifie donc également que les extensions au protocole OAuth 2.0 sont également applicables.

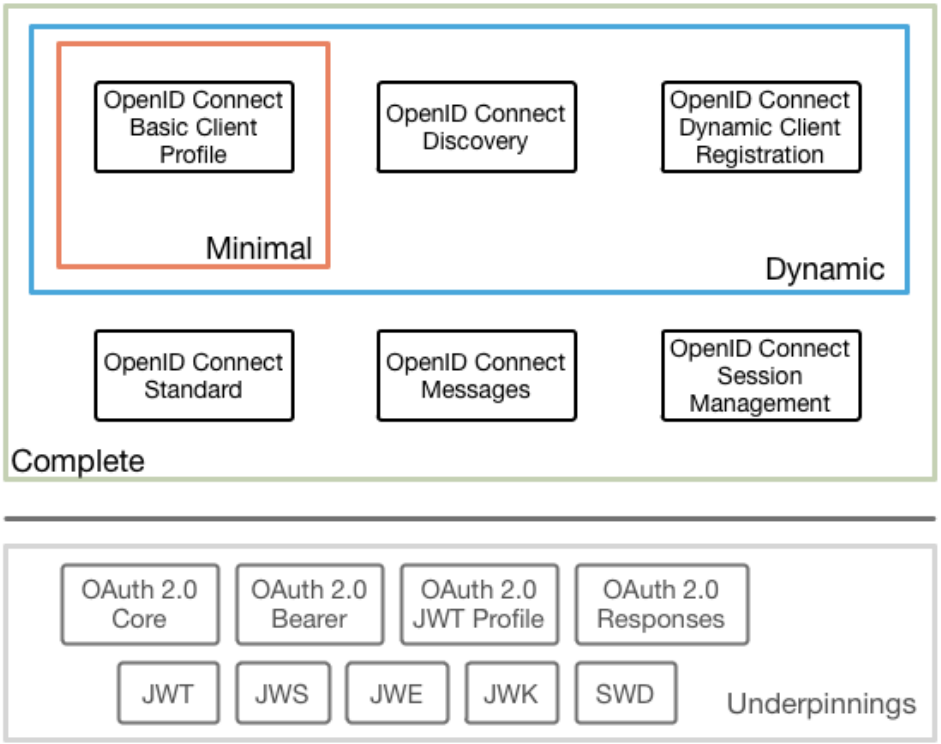


Figure 6-1 : Ensemble de spécifications OpenID Connect 1.0 [Dingle, 2011]

Voici une description succincte des spécifications OpenID Connect reprises au sein de la Figure 6-1 :

- **Basic Client Profile** [Sakimura et al, 2012a] : spécification simple et légère pour un fournisseur de services orienté web. Cette spécification est suffisante pour une majorité des cas basiques. Si l'on s'écarte quelque peu de ce cas de figure, les autres spécifications, et notamment la spécification Standard, devront être consultées ;

- **Discovery** [Sakimura et al, 2011b]: spécification facultative définissant la manière dont les endpoints des utilisateurs et des fournisseurs sont découverts de manière dynamique ;
- **Dynamic Registration** [Sakimura et al, 2012b] : spécification facultative définissant la manière dont les clients s'enregistrent de façon dynamique avec les fournisseurs OpenID ;
- **Standard** [Sakimura et al, 2012c] : spécification de liaisons HTTP, à destination des clients et des fournisseurs OpenID ;
- **Messages** [Sakimura et al, 2012d] : liste tous les messages qui sont utilisés au sein d'OpenID Connect. Cette spécification, facultative, peut être utilisée pour créer des connecteurs supplémentaires ou obtenir des informations complémentaires au sujet d'un message particulier ;
- **Session Management** [Sakimura et al, 2011a] : spécification facultative définissant la façon de gérer les sessions au sein d'OpenID Connect. C'est également au sein de cette spécification que la déconnexion distribuée est décrite ;
- **OAuth 2.0 Multiple Response Type Encoding Practices** [de Medeiros et al, 2011] : document de référence, facultatif, pour définir de nouveaux types de réponses spécifiques, conformément aux stipulations de l' « OAuth Parameters Registry » [Hammer et al, 2012].

Les autres spécifications mentionnées dans le cadre « Underpinnings » de la Figure 6-1 sont des spécifications d'OAuth 2.0. Elles ne seront pas abordées dans ce chapitre mais sont laissées afin de garder une vision d'ensemble de ce qu'est OpenID Connect 1.0. Elles seront également mentionnées dans la section 6.5.

Les documents définissant OpenID Connect 1.0 sont clairs et les différentes requêtes et réponses possibles pour chacune des spécifications sont très bien illustrées en comparaison avec les spécifications précédemment étudiées. Les spécifications de ces différents composants peuvent être retrouvées sur le site officiel[1] d'OpenID Connect.

Ces documents permettent :

- de définir de manière correcte la façon dont les endpoints de jetons et d'autorisation doivent interagir lorsqu'ils authentifient et autorisent les utilisateurs utilisant OAuth v2.0 ;
- de décrire comment les endpoints traditionnels peuvent interagir afin de s'échanger de l'information :
    - au sujet de l'utilisateur possédant le jeton d'accès OAuth ;
    - sur le jeton lui-même ;
- de décrire comment les parties peuvent interagir dynamiquement pour enregistrer un client et comment gérer les sessions au nom d'un utilisateur.

L'étude portera sur les 3 spécifications qui composent le bloc « Dynamic » de la Figure 6-1. En effet, à elles trois, ces spécifications couvrent une grande partie des aspects d'OpenID Connect.

Notons que la spécification OpenID Connect Message définit principalement les endpoints et les formats de messages associés à ceux-ci. Cette spécification peut être très pratique lors de la mise en place d'un des acteurs d'OpenID Connect, elle permet de redéfinir très rapidement chacun des endpoints et requêtes/réponses y étant associés. Elle permet également d'approfondir certains aspects non couverts par d'autres spécifications et possède les références nécessaires pour y arriver. La spécification OpenID Connect Messages, reste donc un document permettant d'approfondir certains aspects des autres spécifications.

[1] http://openid.net/connect/

6.3 Concepts de base

Dans les spécifications abordées tout au long de cette section, nous parlerons principalement du flux d'octroi implicite.

Le flux par code d'autorisation, abordé dans la spécification OpenID Connect Standard, se déduit facilement à l'aide des sections suivantes et de la spécification OpenID Connect Messages.

Les schémas de cette section ne mettent plus en jeu des acteurs mais des endpoints. Ceci permet notamment de pouvoir déployer le endpoint où bon nous semble (avec certaines limites suivant les cas de figure).

La Figure 6-2 représente une vue des interactions entre le fournisseur de services et les différents endpoints pour le flux implicite avec enregistrement dynamique et mécanisme de découverte. L'abréviation « EP » pour « endpoint » a été utilisée afin de rendre le schéma plus lisible. A noter que l'enregistrement dynamique peut se produire à n'importe quel moment du flux (afin de mettre à jour les données enregistrées par exemple).

Le point de vue abordé sur cette figure est la vision du fournisseur de services. Il y a également des interactions qui s'effectuent immédiatement avec l'utilisateur mais cette distinction n'est pas reprise au sein de ce schéma. Si des interactions avec l'utilisateur sont nécessaires à la bonne compréhension du diagramme de séquence, elles seront indiquées par des boucles sur le fournisseur de service (repris sous le terme client). Il en sera de même pour toutes les figures présentes au sein de ce chapitre.

Le jeton ID est utilisé pour gérer l'authentification et l'identifiant de l'utilisateur. Le champ d'application de ce jeton est limité à un client particulier. Il est utilisé comme un jeton d'accès, pour le endpoint CheckID ou il est consommé immédiatement par le client.

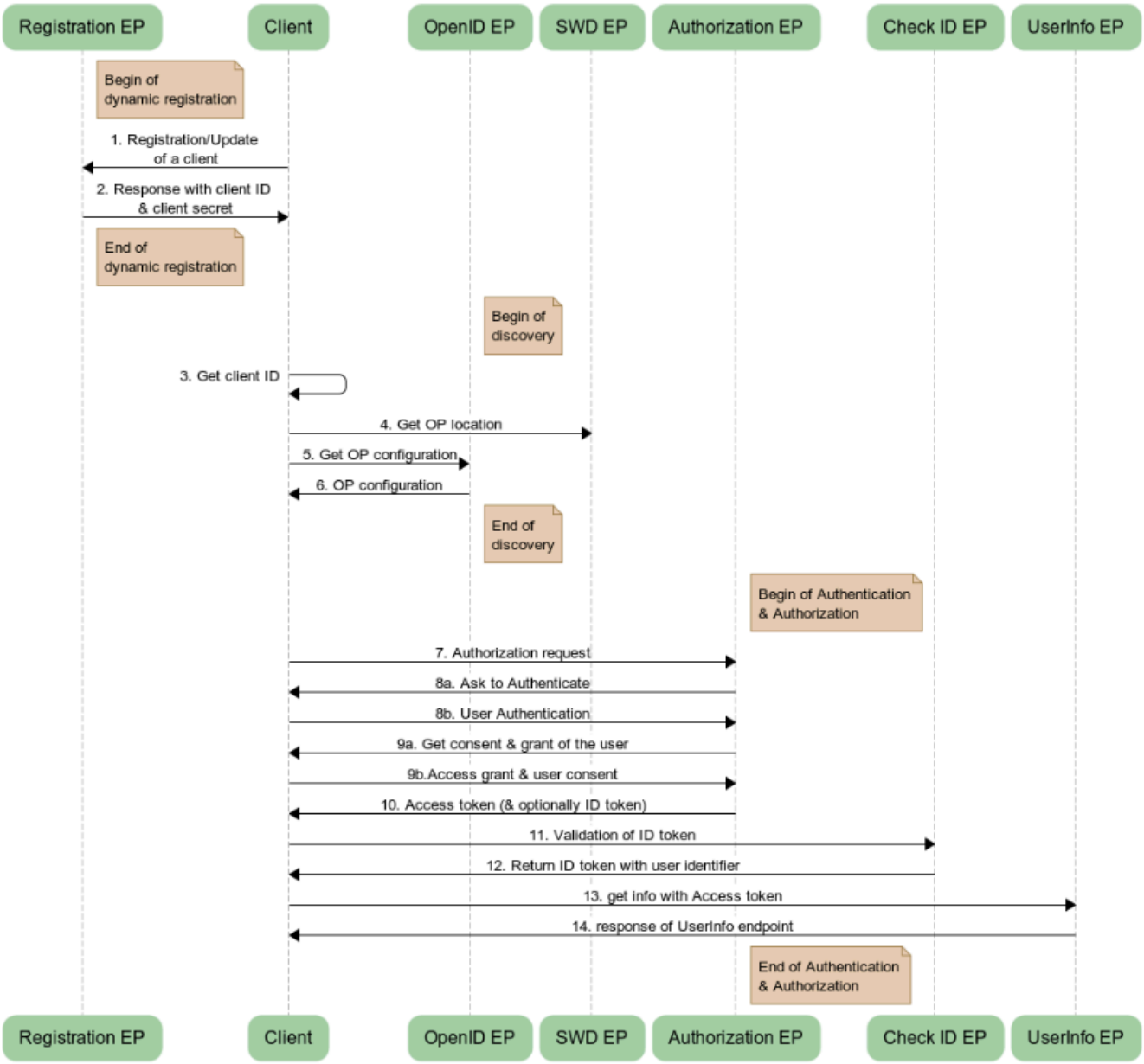


Figure 6-2 : Flux abstrait OpenID Connect 1.0

Les étapes du flux implicite sont les suivantes :

1. Le client envoie une requête d'enregistrement ou de mise à jour du client.
2. Le endpoint Registration répond avec un client_id et un client_secret.
3. Le client obtient l'identifiant de l'utilisateur.
4. Le client effectue une requête vers les endpoints des serveurs SWD (cf. Glossaire) afin de connaître le fournisseur d'OpenID sur base de l'identifiant de l'utilisateur.
5. Le client effectue une requête vers le endpoint du fournisseur OpenID (obtenu à l'étape précédente) afin d'en récupérer sa configuration.
6. Le fournisseur d'OpenID répond à la requête en fournissant un ensemble de paramètres nécessaires (endpoints, champs d'application supportés,…) pour l'étape 7.
7. Le client prépare la requête d'autorisation contenant les paramètres désirés et l'envoie au serveur d'autorisation.
8. Le serveur d'autorisation authentifie l'utilisateur.

9. Le serveur d'autorisation obtient le consentement et l'autorisation de l'utilisateur.
10. Le serveur d'autorisation renvoie l'utilisateur vers le client avec un jeton d'accès et un jeton ID s'il a été demandé.
11. (facultatif) Le client valide le jeton ID sur le endpoint CheckID.
12. (facultatif) Le client reçoit la réponse du jeton ID avec l'identifiant de l'utilisateur.
13. (facultatif) Le client accède au endpoint UserInfo avec le jeton d'accès.
14. (facultatif) Le client reçoit la réponse du endpoint UserInfo avec les données de l'utilisateur.

6.4 Etude du protocole

6.4.1 Profil du client de base

La spécification du profil du client de base, appelé « Basic Client Profile » dans la littérature, implémente OpenID Connect pour le flux implicite et l'intégralité des requêtes appropriées afin d'obtenir l'information souhaitée.

OpenID Connect Basic Client est un des profils de la spécification OpenID Connect Standard 1.0 [Sakimura et al, 2012c] conçue de manière à ce qu'elle soit facilement lisible et facile à mettre en œuvre pour les fournisseurs de services orientés web utilisant le type d'octroi implicite du protocole OAuth 2.0 [Hammer et al, 2012].

Les requêtes d'autorisation peuvent utiliser l'un des deux flux suivants :

- le flux d'octroi implicite : adapté pour les client pouvant maintenir de façon sécurisée un secret client entre eux et le serveur d'autorisation ;
- le flux par code d'autorisation : adapté pour les clients n'étant pas capables de maintenir de façon sécurisée un secret.

Cette spécification ne documente que le flux implicite mais les fournisseurs OpenID doivent supporter les deux flux et se référer à la spécification OpenID Connect Standard 1.0 [Sakimura et al, 2012c] pour le flux par code d'autorisation.

6.4.1.1 Champs d'application

Les clients d'OpenID Connect 1.0 utilisent les valeurs du champ d'application telles que définies dans OAuth 2.0 [Hammer et al, 2012] afin de spécifier les ressources étant disponibles lors de l'utilisation des jetons d'accès.

Pour OpenID Connect, ces champs peuvent être utilisés pour demander un ensemble spécifique d'information depuis le endpoint UserInfo mais également pour faire la demande d'un jeton ID. N'oublions pas qu'OAuth 2.0 permet également de décrire des valeurs de champs d'application additionnelles appelées extensions.

Plusieurs champs d'application peuvent être demandés en créant une liste de valeurs de champ d'application (*scope)* sensibles à la casse et délimitées par des espaces.

Pour augmenter l'activation de nouveaux comptes, un client peut choisir de ne demander qu'un sous-ensemble des informations disponibles sur le endpoint UserInfo.

6.4.1.2 Flux d'octroi implicite

Le schéma de la Figure 6-3 représente le flux implicite décrit dans la spécification du « Basic Client Profile ». L'explication de chacune de ces étapes est reprise sous la figure.

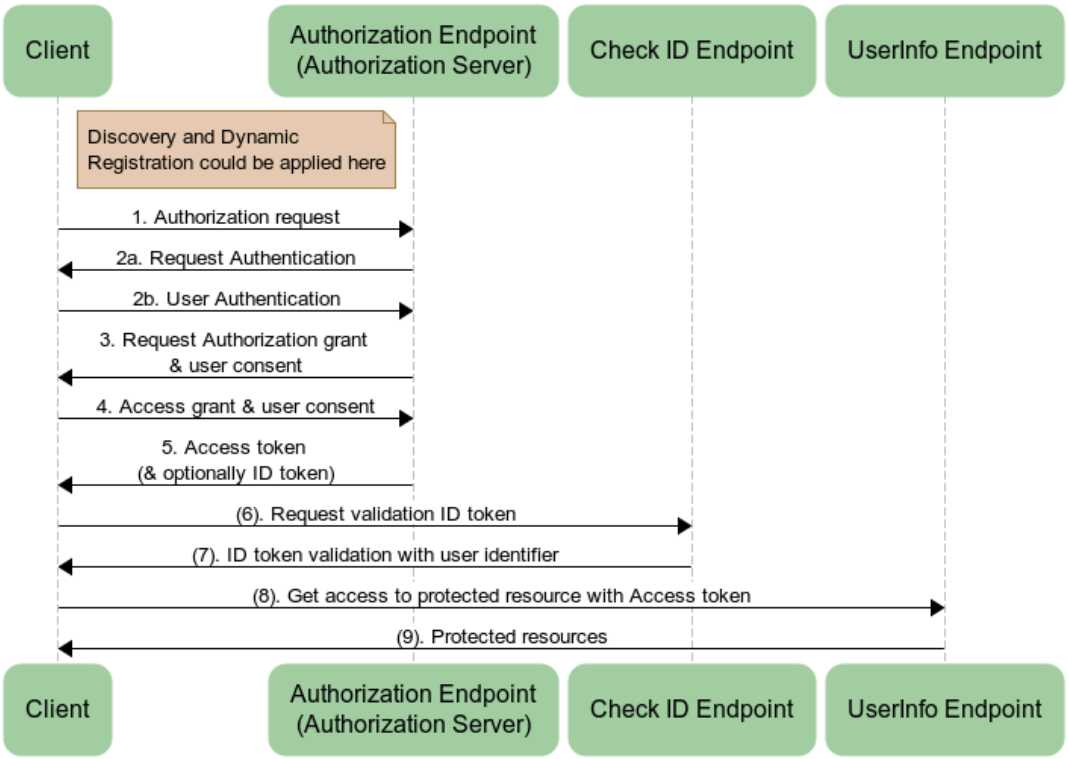


Figure 6-3 : Flux d'octroi implicite OpenID Connect 1.0 (profil du client de base)

1. Le client prépare une requête d'autorisation contenant les paramètres requis et souhaités et l'envoie au serveur d'autorisation via des redirections HTTPS, hyperlink, ou tout autre moyen sécurisé. Cette requête est construite lorsque l'utilisateur désire accéder à une ressource protégée, et que l'autorisation de l'utilisateur n'a pas encore été obtenue.
   Un paramètre peut être utilisé par le client pour s'assurer que l'utilisateur est toujours présent pour la session ouverte ou pour attirer son attention sur la requête.
2. Le serveur d'autorisation authentifie l'utilisateur.
3. Le serveur d'autorisation effectue une demande d'autorisation pour le champ d'application demandé par l'intermédiaire d'une boite de dialogue permettant à l'utilisateur :
   - de reconnaître clairement ce qu'il va approuver ;
   - d'obtenir son consentement.

   La valeur du champ d'application « openid » autorise l'accès du tiers à l'identifiant de l'utilisateur de la session authentifiée. Toutes les autres valeurs de champs d'application sont facultatives.
4. Le serveur d'autorisation obtient le(s) consentement(s)/autorisation(s) de l'utilisateur.
5. Le serveur d'autorisation renvoie :
   - une réponse valide si le propriétaire de la ressource autorise la requête d'accès. Le serveur d'autorisation fournit un jeton d'accès, éventuellement un jeton ID et le(s) délivre(nt) au client. Le client peut alors utiliser le jeton d'accès pour accéder aux ressources protégées sur le serveur concerné ;
   - une erreur si l'utilisateur refuse la demande d'autorisation ou si l'authentification de l'utilisateur a échoué.

6. Le client, s'il le souhaite, effectue ensuite une demande de validité du jeton ID via le endpoint CheckID qui le valide. Cet endpoint est utilisé par les clients n'étant pas capables ou ne souhaitant pas traiter les jetons ID immédiatement.
   Les clients souhaitant traiter immédiatement le contenu du jeton ID doivent consulter la spécification OpenID Connect Standard 1.0 [Sakimura et al, 2012c].
   Le jeton ID est utilisé pour gérer les évènements d'inscription et l'identifiant de l'utilisateur.
7. La réponse CheckID renvoie l'information JSON sérialisée contenue dans le jeton ID ainsi que l'identifiant de l'utilisateur. Afin de vérifier la validité de la réponse, le client doit :
   - vérifier que le endpoint CheckID n'a pas renvoyé d'erreur (jeton ID expiré ou invalide) ;
   - vérifier que le paramètre représentant l'identifiant est équivalent à celui préconfiguré ou obtenu durant le processus de découverte pour la session de l'utilisateur ;
   - la valeur du paramètre *nonce* doit être vérifiée afin de s'assurer que sa valeur est identique à celle qui a été envoyé dans la requête d'autorisation afin de se prémunir contre les attaques par rejeu ;
   - le client doit valider que l'id du client présent dans le paramètre, permettant d'identifier le public à qui ce jeton ID est destiné, est bien celui qui a été enregistré pour la personne identifiée ;
   - l'heure actuelle doit être plus petite que le délai d'expiration à partir duquel (ou après lequel) le jeton ID ne doit plus être accepté ;
   - le paramètre représentant le temps auquel le JWT [Jones et al, 2011b] a été délivré peut être utilisé par le client pour refuser les jetons expirés depuis longtemps. Ceci limite ainsi le temps pendant lequel les paramètres *nonce* doivent être stockés.

   Si l'un des points mentionnés ci-dessus n'est pas respecté, une erreur doit être renvoyée par le endpoint CheckID.
8. Pour obtenir des attributs concernant l'utilisateur, le client effectue une requête vers le endpoint UserInfo. La réponse du endpoint UserInfo n'offre aucune garantie au sujet de l'utilisateur identifié.
9. Si le schéma demandé est « openid », la réponse doit renvoyer un objet JSON [Crockford, 2011] contenant l'ensemble ou un sous-ensemble d'informations concernant l'utilisateur telles que son identifiant, son nom, son prénom, son surnom, son patronyme, l'URL de son blog ou site web, son email, …
   L'endpoint UserInfo doit renvoyer un objet JSON si aucun autre format n'a été spécifié durant l'OpenID Connect Dynamic Client Registration 1.0 [Sakimura et al, 2011b].
   En cas d'erreur, l'endpoint du UserInfo renverra une réponse telle que définie dans la spécification OAuth 2.0 Bearer Tokens[Jones et al, 2012].

6.4.2 Découverte

Cette spécification traite la découverte, appelée « Discovery » dans la littérature, des fournisseurs d'identité. Elle fournit un mécanisme de découverte des fournisseurs OpenID ainsi que des différents endpoints utilisés par la suite de spécifications OpenID Connect.

Certaines installations d'OpenID Connect peuvent utiliser un ensemble de fournisseurs OpenID ou fournisseurs de services configurés au préalable. Dans ce cas, il n'est pas nécessaire de supporter la découverte dynamique d'informations au sujet des identités, services ou enregistrements dynamiques des clients.

Cependant, si certaines implémentations choisissent de supporter des interactions non prévues entre les fournisseurs de services et les fournisseurs OpenID n'ayant pas de

relations préconfigurées, elles devraient accomplir cette tâche en implémentant cette spécification.

OpenID Connect utilise le SWD (Simple Web Discover) [Jones et Goland, 2011] afin de découvrir les différents fournisseurs OpenID. Une fois cette étape effectuée, le endpoint et les autres informations de configuration au sujet du fournisseur OpenID sont retirées sous forme d'un document JSON.

Les différentes étapes reprises dans la spécification « Discovery » et mentionnées ci-dessous sont reprises sur la Figure 6-2 (étapes 3 à 6).

6.4.2.1 Découverte des fournisseurs

Afin de pouvoir démarrer la découverte des endpoints OpenID, l'utilisateur fournit un identifiant au fournisseur de services. Le client applique ensuite des règles de normalisation à l'identifiant. Il effectue, ensuite, une requête HTTPS vers les endpoints des serveurs de SWD afin d'obtenir l'emplacement du service demandé. Le client doit effectuer une vérification du certificat. En réponse, le client reçoit le schéma, le serveur, et éventuellement le port sur lequel le client doit se connecter.

La normalisation

La normalisation consiste à extraire des informations à partir de l'entrée de l'utilisateur. Ces éléments sont utilisés comme entrée au SWD.

L'identifiant de l'utilisateur devrait être une adresse email ou une URL.

Les règles de normalisation de l'identifiant peuvent être étendues par des spécifications complémentaires afin de permettre d'autres types d'identifiants tels que les numéros de téléphone ou des XRI [Reed et McAlpin, 2005].

La normalisation s'effectue différemment suivant qu'elle est appliquée :

- sur des types d'identifiant ;
- sur des adresses email ;
- sur des URL.

La redirection

Si la requête SWD est gérée sur un serveur ou endpoint autre que celui déduit de l'identifiant de l'utilisateur, le serveur retournera un objet JSON [Crockford, 2011] contenant la nouvelle adresse.

Cet objet est composé d'un couple de paramètres avec les valeurs respectives de l'objet JSON et l'URI encodé sous forme d'une chaine de caractères.

Les arguments du formulaire SWD (récupéré à l'étape précédente) sont ensuite utilisés pour construire une requête à destination de l'URI.

6.4.2.2 Information de configuration du fournisseur

L'étape décrite dans cette section est facultative. Les endpoints des fournisseurs OpenID et l'information de configuration peuvent être obtenus par d'autres moyens. Afin de fournir ces informations, les fournisseurs OpenID doivent mettre à disposition un document JSON accessible au client. Ce dernier, effectuera une vérification du certificat du serveur.

Le fournisseur répondra à la requête du client avec un ensemble de paramètres tels que la configuration du fournisseur, les différents endpoints nécessaires, les variables de champs d'application supportées, le lieu où se situe la clé publique, l'emplacement de différents certificats, les algorithmes à utiliser,… Le tout contenu dans un objet JSON.

6.4.3 Inscription dynamique

La spécification d'inscription dynamique, appelée « Dynamic Registration » dans la littérature, décrit comment un fournisseur de services peut acquérir les identifiants requis par la suite de protocoles OpenID Connect.

Si certaines implémentations choisissent de supporter des interactions non prévues entre les fournisseurs de services et les fournisseurs OpenID n'ayant pas de relations préconfigurées, elles devraient accomplir cette tâche en implémentant cette spécification.

Afin de pouvoir offrir la possibilité à un client OpenID d'utiliser les services OpenID, le client doit au préalable s'enregistrer avec le fournisseur OpenID afin d'acquérir un ID client et un secret partagé.

Cette spécification décrit la procédure d'enregistrement d'un nouveau client avec le fournisseur et la procédure de retrait des informations mises à jour pour un client possédant un ID client.

Les différentes étapes reprises dans la spécification « Dynamic Registration » et mentionnées ci-dessous sont reprises sur la Figure 6-2 (étapes 1 et 2).

6.4.3.1 Endpoint d'enregistrement du client

Le endpoint d'enregistrement du fournisseur de service (client) est une ressource protégée OAuth 2.0 qui transmet l'information d'enregistrement au client. Ces informations lui permettront de se configurer pour accéder au fournisseur OpenID.

Le fournisseur OpenID peut exiger un jeton d'accès afin de restreindre les requêtes d'enregistrement uniquement aux clients autorisés.

Afin de pouvoir fournir un support lors de l'enregistrement ouvert, le endpoint d'enregistrement du client doit être capable d'accepter des jetons d'accès tels que décrits dans la spécification OAuth 2.0 Bearer Tokens [Jones et al, 2012].

Les fournisseurs de services doivent envoyer les requêtes avec différents paramètres tels que : le type d'enregistrement (nouvel enregistrement ou mise à jour), les personnes autorisées à administrer l'information sur le endpoint, le type d'application, son nom, le type d'authentification à utiliser pour le endpoint du jeton, le document JWK [Jones, 2011], les emplacements des certificats, l'algorithme JWS [Jones et al, 2011a] à utiliser,… La réponse renvoyée à ces requêtes est un objet JSON.

En cas d'erreur, le endpoint « Client Registration » renverra une erreur comme défini dans la section 3 de la spécification OAuth 2.0 Bearer Token [Jones et al, 2012]. Pour information, cette spécification définit tout de même d'autres codes d'erreurs.

6.5 Considérations en termes de sécurité

Les considérations en termes de sécurité présentes dans la spécification OAuth 2.0 Threat Model and Security Considerations[Lodderstedt et al, 2011] sont évidemment applicables à ce standard. Les recommandations complémentaires suivantes sont également à prendre en considérations.

Divulgation d'une demande

Si les mesures appropriées ne sont pas appliquées, une requête peut être divulguée à un attaquant lui offrant, ainsi, des failles de sécurité et des possibilités d'atteinte à la vie privée. Cette attaque fonctionne avec un user-agent compromis dans le cas de requête indirecte.

Cette spécification fournit un moyen d'offrir la confidentialité de la requête de bout en bout puisque le contenu de la requête est encrypté en JWT avec une clé et un chiffrement approprié.

« Déguisement » du serveur

Un serveur malintentionné peut se faire passer pour un serveur légitime par différents moyens. Pour détecter ce genre d'attaque, le client doit s'authentifier avec le serveur. Ce standard fournit également un moyen de s'authentifier avec le serveur à travers l'utilisation de JWT signée (JWS) ou de JWT chiffrée (JWE) avec la clé et le chiffrement appropriés.

Fabrication/modification de jeton

Une personne malintentionnée peut générer un faux jeton ou modifier le contenu d'un jeton existant. Cet acte a pour conséquence d'autoriser des accès inappropriés à l'utilisateur de la part du fournisseur de services. Par exemple :

- une personne malintentionnée peut modifier le jeton pour étendre sa période de validité ;
- un client peut modifier le jeton analysé pour avoir accès à une information à laquelle il ne devrait pas avoir accès.

Il y a deux façons de réduire ce type d'attaque :

- le jeton peut être signé par le fournisseur OpenID. Le fournisseur de services devrait vérifier la signature afin de vérifier que celle-ci a bien été fournie par un fournisseur OpenID légitime ;
- le jeton peut être envoyé sur un canal sécurisé tel que TLS/SSL. Afin de protéger l'intégrité du jeton d'une attaque malicieuse, le serveur doit être authentifié.

Divulgation de la réponse du serveur

La réponse du serveur peut contenir l'authentification et des déclarations d'attribut contenant une information sensible relative au fournisseur de services. La divulgation du contenu de la réponse peut rendre le client vulnérable à d'autres types d'attaque.

La divulgation de la réponse du serveur peut être atténuée des deux façons suivantes :

- en utilisant le code d'autorisation. La réponse est envoyée sur un canal protégé par SSL/TLS, où le client est authentifié par l'ID du fournisseur de services et son secret ;
- pour les autres types de réponses, la réponse signée peut être encryptée avec la clé publique du client ou un secret partagé comme une JWT chiffrée avec une clé et un chiffrement appropriés.

Répudiation de la réponse du serveur

Une réponse peut être répudiée par le serveur si les mécanismes appropriés ne sont pas mis en place. Par exemple, si un serveur ne signe pas une réponse, le serveur peut déclarer qu'elle n'a pas été générée par les services dudit serveur.

Pour atténuer cette menace, la réponse peut être signée par le serveur en utilisant une clé supportant la non-répudiation. Le client devrait vérifier que la signature a bien été fournie par un serveur légitime et que l'intégrité est bien respectée.

Répudiation de la requête

Depuis qu'il est possible qu'un fournisseur de services compromis ou malintentionné envoie des requêtes vers un mauvais tiers, un fournisseur de services qui était authentifié en utilisant seulement les types de jetons « bearer » peut rejeter toute transaction.

Pour atténuer cette menace, le serveur peut exiger que la requête utilise une clé supportant la non-répudiation. Le serveur devrait vérifier la signature pour vérifier qu'elle a été fournie par un fournisseur de services légitime et que l'intégrité a bien été respectée.

Redirection des jetons d'accès

Une personne malintentionnée utilise le jeton d'accès généré pour une ressource afin d'obtenir l'accès à une seconde.

Pour atténuer cette attaque, le jeton d'accès devrait être restreint au niveau du public ciblé et de son champ d'application. Une façon de mettre cela en place est d'introduire l'identifiant de la ressource du public pour lequel le jeton a été généré.

La ressource vérifie que les jetons entrants contiennent bien son identifiant en tant que clé publique du jeton.

Attaque par rejeu

Une personne malintentionnée essaye d'utiliser un jeton à utilisation unique qui aurait été utilisé une fois avec la ressource prévue.

Pour atténuer ce type d'attaque, le jeton devrait introduire un horodatage et une durée de vie très courte en termes de validité. Le fournisseur de services vérifie les valeurs de l'horodatage et de la durée de vie afin de s'assurer que le jeton est valide au moment de la demande.

Le serveur peut, éventuellement, enregistrer l'état du jeton et vérifier le statut de chaque requête.

Ecoute du réseau

En plus des patterns d'attaque décrits dans OAuth 2.0 Threat Model and Security Considerations [Lodderstedt et al, 2011] (section 4.4.1.1), le code d'autorisation peut, si le user-agent est infesté par un malware, être capturé dans le user-agent puisque la session TLS s'y termine.

Par contre, sa capture n'est utile que si le profil n'utilise pas l'authentification cliente ou le chiffrement de la réponse.

Substitution de jeton

Un utilisateur peut essayer d'usurper l'identité d'un autre utilisateur avec plus de privilèges en renversant le canal de communication entre le endpoint du jeton et du client. Il pourrait, par exemple, réordonner les messages envoyés, en tentant de convaincre le endpoint du jeton que son octroi d'autorisation correspond à celui envoyé au nom de l'utilisateur avec plus de privilèges.

Les réponses des requêtes du jeton sont liées avec les requêtes HTTP correspondantes par ordre de message. Aussi bien les jetons que les requêtes sont protégées par TLS. Ceci permet de détecter et refuser le réordonnancement non souhaité des paquets.

Attaque temporelle

Les implémentations ne devraient pas terminer le processus de vérification au moment où elles trouvent une erreur mais devraient continuer à fonctionner jusqu'à ce que tous les octets aient été traités pour éviter ce type d'attaque.

Cryptanalyse

Il y a différentes variantes possibles aux attaques liées à la cryptographie. Elles dépendent de la méthode utilisée pour le chiffrement et de la signature / vérification de l'intégrité.

Les implémentations devraient prendre en compte les considérations en termes de sécurité concernant JWS, JWE, JWT pour éviter les vulnérabilités liées.

Ordre de chiffrement et signature

Les signatures sur des textes chiffrés ne sont pas considérées comme valides dans plusieurs juridictions. Pour l'intégrité et la non-répudiation cette spécification exige la signature de texte brut en JSON.

Les implémentations devraient consulter les considérations de sécurité concernant JWE pour éviter cette vulnérabilité.

Fournisseur d'identité

OpenID Connect 1.0 supporte un seul fournisseur d'OpenID par combinaison serveur/port. Il est recommandé qu'un seul fournisseur par serveur soit donc utilisé. Le

SWD traite le composant du chemin de tout URI comme une partie de l'identifiant de l'utilisateur.

Exigences TLS

Les implémentations doivent supporter TLS. La version devant être mise en œuvre peut varier au fil du temps, et dépend de l'étendue du déploiement et des vulnérabilités de sécurité connues au moment de l'implémentation effective.

A l'heure d'écrire ces lignes, la version TLS 1.2 est la version la plus récente, mais est très limitée en termes de déploiement, et donc, peut ne pas être facilement disponible dans les « boites à outils ». TLS en version 1.0 est la version la plus largement déployée. Cette version donnera de plus larges interopérabilités.

Pour se protéger contre la divulgation d'informations et la falsification, la protection de la confidentialité doit être appliquée en utilisant TLS comme une suite d'algorithmes de chiffrement fournissant une protection de la confidentialité et de l'intégrité.

Requêtes vers les endpoints

Les requêtes vers le endpoint Client Registration transmettent en texte clair certains identifiants dans la requête et la réponse. Pour cette raison, le serveur doit exiger l'utilisation d'un mécanisme de sécurité pour envoyer les requêtes au endpoint Registration. Le serveur doit être compatible avec TLS et peut également supporter d'autres mécanismes de couches de transport rencontrant les exigences de sécurité. Lorsque TLS est utilisé, le client doit effectuer une vérification du certificat.

Les requêtes à destination du endpoint Registration pour la mise à jour des informations du client doivent s'effectuer en nombre limité lorsque celles-ci échouent, afin d'éviter d'exposer de trop nombreuses fois le secret du client via des essais d'accès répétés.

Si le serveur d'autorisation supporte l'enregistrement ouvert d'un client, on doit être extrêmement attentif à toutes URL fournies par le client qui seraient affichées par l'utilisateur. Un client malintentionné pourrait former une requête avec une référence vers une politique d'URL forçant le téléchargement.

6.6 Respect de la vie privée

La réponse du endpoint UserInfo contient habituellement des informations permettant d'identifier une personne. Pour cette raison, le consentement de l'utilisateur lors la divulgation de cette information devrait être obtenue à l'autorisation ou avant (conformément à la réglementation applicable). L'objectif recherché lors de l'utilisation de ces informations est enregistré avec le paramètre *redirect_uri*.

Seules les informations nécessaires devraient être stockées sur le client, et celui-ci devrait associer les données reçues avec la raison pour laquelle elles ont été demandées.

Le serveur possédant la ressource devrait mettre à disposition de l'utilisateur les journaux d'évènements afin que ce dernier puisse également surveiller les personnes accédant à ses données, agissant ainsi comme un contrôleur.

Finalement et afin de protéger l'utilisateur d'une corrélation entre les différents clients, l'utilisation de paires d'identificateurs pseudonymes (PPID) tels que l'ID de l'utilisateur devrait être prise en considération.

Durée de vie des jetons de rafraichissement et jeton d'accès

Les permissions données au jeton d'accès ne peuvent être révoquées par le serveur d'autorisation. La permission du jeton d'accès devrait être à usage unique et d'une durée de vie très limitée.

Si l'accès au endpoint UserInfo ou à d'autres ressources protégées est nécessaire, le jeton de rafraichissement devrait être utilisé. Le client peut alors échanger le jeton de

rafraichissement sur le endpoint du jeton contre un jeton d'accès de courte durée. Ce dernier pourra être utilisé pour accéder à la ressource.

Le serveur d'autorisation devrait clairement identifier les autorisations à long terme données à l'utilisateur pendant l'autorisation.

Le serveur d'autorisation doit offrir un mécanisme à l'utilisateur pour qu'il puisse révoquer les jetons de rafraichissement donnant autorité à un client.

L'email comme identifiant de l'utilisateur

Rappelons que certaines publications [Lei et al, 2010] semblent pointer du doigt les modules d'authentification utilisant les adresses mail comme identifiant unique comme un danger. Il est donc également important de protéger cette information utilisée comme base pour la découverte des fournisseurs OpenID depuis le fournisseur de services.

6.7 Synthèse

OpenID Connect 1.0 est une spécification en cours de révision bien prometteuse. Les attributs de celle-ci allient les fonctionnalités d'OpenID 2.0 avec la facilité de sa mise en place au dessus d'OAuth 2.0, standard fiable déjà adopté par de nombreux acteurs.

L'intégration d'OpenID avec OAuth 2.0 était jusqu'ici relativement compliquée et peu documentée. Ces éléments ont rebuté plusieurs acteurs au vu des risques non négligeables liés à une mauvaise implémentation. Cette spécification devrait faciliter et permettre à ces personnes de se lancer dans son intégration. Elle offre, en effet, un guide clair, et une spécification adaptée aux besoins suivant l'utilisation que l'on souhaite en faire.

Le seul reproche, que l'on pourrait faire à son égard, est son manque de clarté entre les différentes spécifications, certainement dans l'optique où celles-ci sont modulaires. En effet, il manque cette approche globale concernant les spécifications à utiliser, une sorte de guide permettant de connaitre l'ensemble de spécifications nécessaires pour un type d'implémentation particulier ainsi que différents schémas d'interaction.

OpenID Connect offre de nombreuses possibilités d'extensions et de définitions de messages complémentaires. De plus, et en comparaison avec les spécifications précédentes, celles d'OpenID Connect sont claires et truffées d'exemples en tous genres rendant difficilement possible les mauvaises interprétations de la spécification.

OpenID Connect semble être moins laxiste en matière de sécurité tout en restant suffisamment souple que pour laisser la possibilité d'intégrer d'autres systèmes. Il permet d'inclure des mécanismes de chiffrement et de signature plus robustes. La spécification permettra également aux participants de supporter d'éventuels chiffrements de données d'identité.

Le mécanisme de découverte, quant à lui, peut fonctionner avec une adresse email. Ceci permettra sans doute, à ce futur standard, d'être plus facilement adopté. L'URL utilisée au sein d'OpenID 2.0 était, en effet, peu user-friendly.

L'aspect anonymat qu'offrait cette URL était un frein à son intégration car ce qui est actuellement recherché par l'ensemble des fournisseurs de services est, avant tout, l'amélioration de l'expérience utilisateur. Pour ce faire, un maximum d'informations au sujet de l'utilisateur est nécessaire. OpenID Connect améliore également ce point en offrant un large panel d'informations concernant l'utilisateur tout en laissant le contrôle à l'utilisateur de ce qu'il souhaite divulguer.

OpenID Connect est donc un standard très prometteur mariant sécurité, robustesse, possibilités d'intégration via extensions, service de découverte, contrôle sur la divulgation, gestion de sessions avancées, partages d'informations….

7 SAML 2.0

7.1 Introduction

SAML sont les initiales de « Security Assertion Markup Language » signifiant littéralement langage de balisage d'assertion de sécurité. SAML est un standard utilisé pour échanger les informations d'identité entre deux domaines.

Il a été créé par OASIS (Organization for the Advancement of Structured Information Standards) et est devenu un standard, en Novembre 2002, dans sa version 1.0. Cette première version a été suivie, un peu moins d'un an plus tard, d'une version 1.1. Le standard est actuellement en version 2.0 depuis Mars 2005, version qui a vu le jour suite à son implémentation en masse par différentes entreprises.

La principale différence vis-à-vis de sa version 1.1[1] concerne l'unification des composants de la fédération d'identité et des adaptations provenant d'initiatives telles que Shibbolet 1.3[2] et Liberty ID-FF 1.2[3] (base de SAML 2.0).

Les avantages de SAML 2.0 sont multiples :

- grandes possibilités d'interopérabilité entre différents modules d'authentification unique ;
- plate-forme neutre : sécurité indépendante de la logique de l'application ;
- pas de couplage : aucune exigence en termes de maintenance et de synchronisation de l'information concernant l'utilisateur ;
- améliorer l'expérience utilisateur en la rendant paramétrable et respectueuse de la vie privée ;
- réduction des coûts administratifs pour les fournisseurs de services (transférés au fournisseur d'identité) ;
- possibilité de transfert des risques de la gestion d'identité vers le fournisseur d'identité ;
- possibilité de sécurisation de web service.

SAML permet de définir :

- un format de message ;
- un protocole de communication ;
- un moyen d'échanger des informations d'authentification, d'autorisation ainsi que des attributs de manière sécurisée entre différents partenaires sur base de la technologie XML.

L'authentification unique, dans le cadre de SAML 2.0, permet de se rapprocher de la fédération d'identité. En effet, SAML 2.0 fournit un moyen, pour les partenaires, de se mettre d'accord et d'établir un identifiant commun et partagé. Cet identifiant est utilisé pour référencer un utilisateur, à la place de partager les informations le concernant, à travers les frontières de l'organisation. Il s'agit donc principalement d'une approche coopérative. Pour cette raison, SAML est utilisé dans des domaines complexes composés de plusieurs sous-domaines, comme par exemple l'Etat et ses différentes institutions.

Seul l'aspect authentification unique de SAML 2.0 et certains services gravitant autour de celui-ci seront abordés.

[1] http://www.oasis-open.org/standards#samlv1.1

[2] http://shibboleth.net/

[3] http://projectliberty.org/resource_center/specifications/liberty_alliance_id_ff_1_2_specifications/?f=resource_center/specifications/liberty_alliance_id_ff_1_2_specifications

7.2 Type et composition

Les acteurs repris dans SAML 2.0 et dans le cadre de ce mémoire sont :
- le fournisseur d'assertion (asserting party) : génère des assertions de sécurité et est comparable au fournisseur d'identité ;
- le fournisseur de services (relying party) : utilise les assertions reçues ;
- l'utilisateur : représenté par un navigateur web.

Pour commencer, nous allons d'abord, et de façon succincte, parcourir les deux cas généraux d'utilisation de SAML 2.0 à savoir:
- l'authentification unique via web consiste en un partenariat entre deux sites web qui se sont mis d'accord sur une identité fédérée. Le service d'identité fournira une assertion au fournisseur de services partenaire attestant que l'utilisateur est connu au sein de son système. Cette assertion sera également accompagnée de toute une série d'attributs d'identité ;
- la fédération d'identité permet d'utiliser l'identité fédérée de l'utilisateur dans les assertions SAML et de la propager parmi tous les fournisseurs de services. Il est donc possible avec SAML d'établir une identité fédérée pour un utilisateur durant l'authentification unique via web. L'identité locale du fournisseur de services sera liée avec son identité fédérée sous forme d'un identifiant commun aux deux acteurs.
  Deux atouts ont été ajoutés à SAML 2.0 pour la fédération d'identité :
  - nouvelles constructions et messages pour l'établissement dynamique d'identités fédérées ;
  - deux nouveaux types d'identifiants ont été introduits pour préserver la vie privée.

Tout ceci est possible grâce à la large variété de spécifications et d'extensions. Les principales d'entre elles sont reprises sur la Figure 7-1.

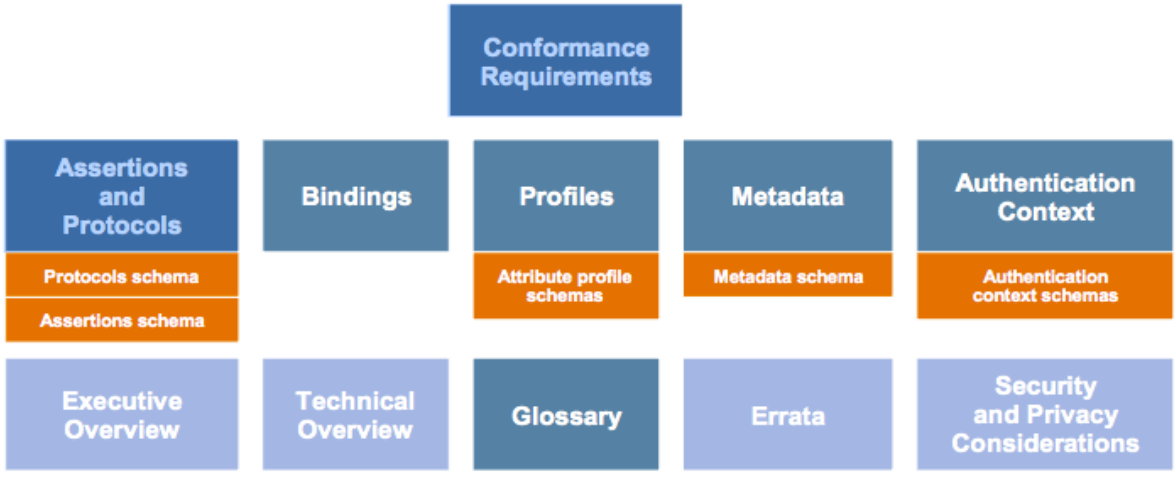


Figure 7-1 : Ensemble de spécifications et extensions de SAML 2.0 [Ragouzis et al, 2008]

Comme on peut le constater, la documentation de SAML 2.0 est conséquente. A titre indicatif, voici sans doute les éléments les plus intéressants :
- glossaire SAML 2.0[1] (les notions liées à SAML y sont expliquées) ;
- un aperçu technique de SAML 2.0 [Ragouzis et al., 2008] ;
- un aperçu de son exécution [Madsen et al, 2005] ;
- exigence de conformité [Mishra et al, 2005].

[1] http://docs.oasis-open.org/security/saml/v2.0/saml-glossary-2.0-os.pdf

7.3 Concepts de base

SAML 2.0 repose sur différents concepts de base qui sont représentés avec leur niveau d'imbrication sur la Figure 7-2.

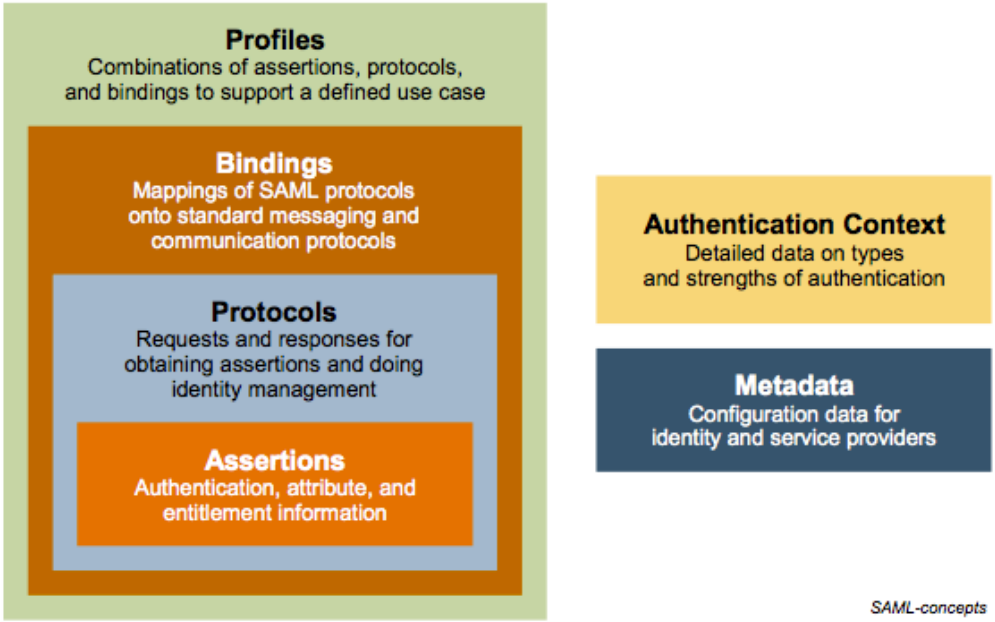


Figure 7-2 : Schéma de l'imbrication des concepts de base de SAML 2.0 [Ragouzis et al, 2008]

SAML dans sa configuration minimale est défini en termes d'assertions, de protocoles, de liaisons, et de profils. Les concepts de contexte d'authentification et de métadonnées sont également deux spécifications importantes mais non indispensables dans le cadre de ce mémoire, leur existence ne sera que brièvement mentionnée.

Les 4 concepts primordiaux de SAML 2.0 sont repris dans les trois spécifications suivantes : SAML Core 2.0[1] (détails techniques), SAML Bindings 2.0 [Cantor et al., 2005], SAML Profiles 2.0 [Hughes et al., 2005].

Nous allons, maintenant parcourir chacun des concepts de base de SAML 2.0 afin de nous familiariser avec eux.

7.3.1 Assertions

Une assertion est un bloc d'informations fournissant une ou plusieurs déclarations, faite(s) par une autorité SAML, considérée(s) comme vraie(s).

SAML définit 3 types d'assertions :

- les assertions d'authentification : spécifiant que le sujet de l'assertion a bien été authentifié et comment il l'a été ;
- les assertions d'attribut : spécifiant les attributs d'identité pour le sujet de l'assertion ;
- les assertions d'autorisation : spécifiant que le sujet de l'assertion possède certaines permissions ;

Voici un exemple d'assertion, sous forme textuelle, d'un fournisseur d'identité :

« Cet utilisateur est Vincent Vermeren, son adresse email est vermeren@gmail.com, et il a été authentifié dans ce système en utilisant le mécanisme par mot de passe ».

Le fournisseur de services choisira d'utiliser ces informations ou pas sur base de sa politique d'accès pour donner accès aux ressources protégées.

[1] http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

Cette information sera, bien évidemment, représentée sous une forme structurée propre à SAML 2.0.

7.3.2 Protocoles

Un protocole définit la structure et le contenu des assertions.
SAML définit différents protocoles permettant aux fournisseurs de services de :

- demander à une autorité SAML une ou plusieurs assertions ;
- demander qu'un fournisseur d'identité authentifie un donneur d'ordre et qu'il renvoie l'assertion correspondante ;
- demander d'enregistrer un identifiant ;
- demander d'arrêter l'utilisation d'un identifiant ;
- retirer un message de protocole qui a été demandé à l'aide d'un artefact (référence vers un message SAML, décrit en détail dans la section suivante);
- effectuer une déconnexion unique ;
- demander une correspondance à un identifiant.

Les différents protocoles de communication pouvant être utilisés au sein de SAML 2.0 sont les suivants :

- **Authentification Request Protocol** : définit un moyen par lequel un « principal » peut demander des assertions contenant des déclarations d'authentification et éventuellement des attributs. Le profil « Web Browser SSO » utilise ce protocole lorsqu'il redirige un fournisseur de services vers un fournisseur d'identité quand il a besoin d'une assertion pour établir un contexte d'authentification sécurisé ;
- **Single Logout Protocol :** définit un mécanisme de déconnexion unique initié par le fournisseur d'identité, l'utilisateur ou le fournisseur de services ;
- **Assertion Query and Request Protocol :** définit un ensemble de requêtes permettant d'obtenir des assertions SAML ;
- **Artifact Resolution Protocol** : définit un mécanisme par lequel les messages SAML peuvent être passés par référence en utilisant un artefact. Un artefact est une valeur petite et de taille fixe pointant vers un message SAML de taille variable et généralement plus grande. Le protocole permet de déréférencer l'artefact de son créateur et de retourner le protocole actuellement utilisé pour les messages ;
- **Name Identifier Management Protocol :** définit un mécanisme pour changer la valeur et le format d'un identifiant utilisé pour référencer un principal. Il permet également de clôturer une association sur un identifiant entre un fournisseur d'identité et un fournisseur de services ;
- **Name Identifier Mapping Protocol :** définit un mécanisme pour établir une relation entre un identifiant SAML et un autre.

7.3.3 Liaisons

La correspondance des échanges de requêtes/réponses, dans la messagerie standard ou les protocoles de communication, est appelée liaison. Une liaison explique comment les messages de protocoles et les moyens de communication bas-niveaux sont utilisés pour transporter les messages SAML entre différents acteurs. Par exemple, la spécification de liaisons SOAP permet de définir comment les messages de protocoles peuvent être communiqués via des messages SOAP alors que la spécification de liaisons HTTP définira comment faire passer des messages par le biais de redirection HTTP.
Les différentes liaisons existantes sont les suivantes :

- **HTTP Redirect Binding :** définit la manière de transporter les messages SAML en utilisant des messages de redirection HTTP ;

- **HTTP POST Binding :** définit la manière de transporter les messages SAML à travers un formulaire HTML ;
- **HTTP Artifact Binding :** définit la manière de transporter un artefact en utilisant HTTP. Deux mécanismes sont définis : un formulaire HTML ou une chaine de caractères au sein de l'URL ;
- **SAML SOAP Binding :** définit la manière de transporter les messages SAML dans des messages SOAP 1.1. Certains détails d'utilisation de SOAP sur HTTP sont également présents ;
- **Reverse SOAP Binding :** définit un échange de messages SOAP/HTTP à plusieurs niveaux. Cela permet à un client HTTP d'agir comme un répondeur SOAP ;
- **SAML URI Binding :** définit un moyen de retirer une assertion SAML existante en résolvant un URI.

7.3.4 Profils

Un profil SAML définit des contraintes et/ou extensions venant supporter l'utilisation de SAML pour une application particulière. Le but recherché est d'améliorer l'interopérabilité, en supprimant certains aspects de flexibilité inévitables, dans des standards à utilisation générale. Par exemple, le profil Web Brower SSO spécifie comment les assertions d'authentification SAML sont communiquées entre le fournisseur d'identité et le fournisseur de services afin de permettre l'authentification unique pour un utilisateur via son navigateur.

Les différents profils existants sont :

- **Web Browser SSO Profile :** définit comment les entités SAML utilisent les réponses SAML, l' « Authentication Request Protocol » et les assertions pour effectuer l'authentification unique dans les navigateurs web standards. Il définit comment les messages sont utilisés en combinaison avec les liaisons HTTP Redirect, HTTP POST et HTTP Artifact ;
- **Enhanced Client and proxy :** définit un profil d'authentification unique dans lequel les clients spécialisés peuvent utiliser les liaisons Reverse-SOAP et SOAP ;
- **Identity Provider Discovery Profile :** définit un mécanisme permettant à un fournisseur de services de prendre connaissance des autres partenaires qu'un utilisateur a visités précédemment ;
- **Single Logout Profile :** définit comment le protocole Single Logout peut être utilisé avec les liaisons SOAP, HTTP Redirect, HTTP POST et HTTP Artifact ;
- **Assertion Query/Request Profile :** définit comment les entités SAML peuvent utiliser le protocole Query and Request afin d'obtenir des assertions SAML sur des liaisons synchrones telles que SOAP ;
- **Artifact Resolution Profile :** définit comment les entités SAML peuvent utiliser le protocole Artifact Resolution sur une liaison synchrone telle que SOAP afin d'obtenir le message faisant référence à un artefact ;
- **Name Identifier Management Profile :** définit comment le protocole Name Identifier Management peut être utilisé avec des liaisons SOAP, HTTP Redirect, HTTP POST et HTTP Artifact ;
- **Name Identifier Mapping Profile :** définit comment le protocole Name Identifier Mapping utilise une liaison synchrone telle que SOAP.

7.3.5 Metadata

La spécification Metadata définit un moyen d'exprimer et de partager l'information de configuration (rôles opérationnels, les liaisons et attributs supportés, l'information sur les clés de chiffrement,…) entre les acteurs SAML. Il est défini lui-même par son propre schéma XML.

7.3.6 Contexte d'authentification

La spécification « contexte d'authentification » permet de définir le contexte d'authentification dans lequel l'utilisateur a été authentifié par un fournisseur d'identité. Cette information est transmise par une assertion d'authentification. Elle permet aux fournisseurs de services d'obtenir des informations sur l'authentification, elle permet également à un fournisseur de services de demander à un fournisseur d'identité d'utiliser une certaine méthode d'authentification pour un utilisateur.

7.3.7 Liens entre les différents composants

La Figure 7-3 représente un message SAML et ses composants, tant internes qu'externes (protocole de transport).

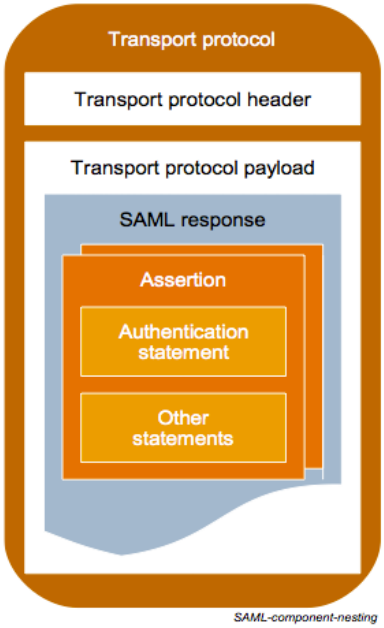


Figure 7-3 : Relations entre les composants SAML 2.0 [Ragouzis et al, 2008]

L'assertion, représentée sous le format XML contiendra, dans l'ordre, une série d'informations concernant la nature de l'assertion (la version de SAML utilisée, celui qui l'a créée, celui qui l'a délivrée), le sujet de l'assertion (à qui le contenu est destiné), la période de validité de l'assertion suivie finalement de la déclaration de l'autorisation.

Une assertion contient une ou plusieurs déclarations ainsi que différentes informations basiques pour chacune d'entre elles et pour l'assertion dans son ensemble.

7.4 Etude du protocole

Afin de réaliser l'étude de ce standard et au vu du grand nombre de spécifications, seuls les profils suivants seront abordés:

- « Web Browser SSO » : profil d'authentification unique via navigateur web ;
- « Identity Provider Discovery » : profil de découverte des fournisseurs d'identité ;
- « Single Logout » : profil de déconnexion unique.

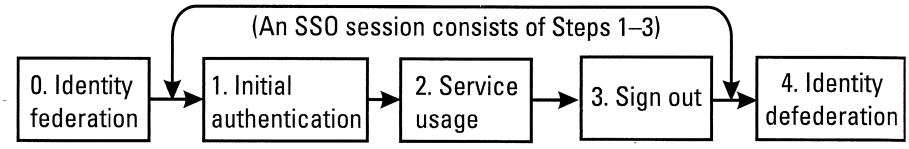
Deux types de mise en œuvre de profil d'authentification unique via web sont présentés dans cette section. Les liaisons utilisées dans ces scénarios sont principalement les liaisons « HTTP Redirect Binding » et « HTTP POST Binding ».

Notons également que la spécification « SAML Core 2.0 » ne sera pas abordée. Elle est un peu le pendant de la spécification « Messages » d'OpenID Connect car elle contient tous les détails techniques pour former un type de message précis.

Nous nous concentrerons particulièrement sur les différents profils et expliquerons certains aspects des liaisons et protocoles concernés au sein de chacun de ceux-ci. Ces différents profils peuvent bien évidemment s'imbriquer et être utilisés dans un même flux.

7.4.1 Fédération d'identité

Avant de découvrir les aspects des différents profils, vous découvrirez sur la Figure 7-4, un schéma représentant le cycle d'une session d'authentification unique dans une fédération d'identité. La fédération d'identité a déjà été présentée partiellement dans la section 7.2. Notons qu'une identité fédérée est une identité qui est à la fois portable et pouvant être transportée et consumée à travers différents domaines autonomes ou frontières d'entreprises.



Step 0: Identity federation—User federates accounts (as needed)
Step 1: Initial authentication—User authenticates for single sign-on
Step 2: Service usage—User accesses services (log on procedure not needed)
Step 3: Sign out—User closes SSO session with single procedure (i.e., single logout)
Step 4: Identity defederation—User defederates accounts (as needed)

Figure 7-4 : Cycle d'une session d'authentification unique SAML 2.0 [Bertino et Takahashi, 2011]

La fédération d'identité diffère de l'authentification unique par les étapes 0 et 4 de la Figure 7-4 représentant respectivement la fédération du compte de l'utilisateur par le fournisseur de services et l'étape inverse.

7.4.2 Profil d'authentification unique via navigateur web

Ce profil est sans doute l'un des plus répandus. Différents flux sont possibles et imaginables. Voici les 3 scénarios les plus répandus de l'authentification unique :

- flux initié par le fournisseur de services. Il utilise une liaison de redirection vers le fournisseur d'identité pour la requête d'authentification et une liaison de type POST pour la réponse depuis le fournisseur d'identité vers le fournisseur de services ;
- flux initié par le fournisseur de services. Il utilise une liaison de type POST pour la requête d'authentification et une liaison de type Artefact pour la réponse ;
- flux initié par le fournisseur d'identité. Il utilise une liaison de type POST pour la réponse depuis le fournisseur d'identité vers le fournisseur de services, aucune requête d'authentification n'est impliquée dans ce scénario.

Nous n'aborderons pas le deuxième cas d'utilisation utilisant les artefacts. Il faudrait, en effet, expliquer la notion de message SOAP non repris dans le cadre de cette étude de SAML.

Le flux initié par le fournisseur de services avec liaison de redirection est représenté à la Figure 7-5 et est décrit à la suite de celui-ci.

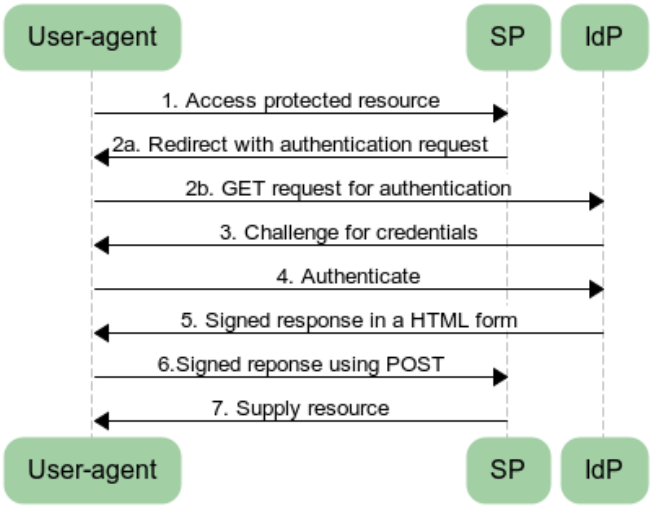


Figure 7-5 : Flux SAML 2.0 initié par le fournisseur de services avec liaisons redirection/POST

1. L'utilisateur tente d'accéder à une ressource sur le fournisseur de services. Ce dernier enregistre la ressource demandée et la transmettra dans la requête à l'étape 2.
2. Le fournisseur de services envoie une réponse de redirection HTTP contenant la destination du service d'authentification unique chez le fournisseur d'identité simultanément avec la requête d'authentification encodée.
   Le navigateur (user-agent sur le graphe) effectue la redirection et envoie donc une requête GET vers l'URL fournie avec la ressource demandée (étape 2b).
3. Le fournisseur d'identité vérifie que l'utilisateur possède un contexte sécurisé d'authentification rencontrant le type d'authentification souhaité ou celui utilisé par défaut. Si c'est le cas, il se rend à l'étape 6 autrement le fournisseur d'identité demande, via le user-agent, des identifiants valides à l'utilisateur (voir étapes 4 & 5).
4. L'utilisateur s'authentifie en fournissant des identifiants valides et un contexte sécurisé d'authentification est créé, localement, pour l'utilisateur chez le fournisseur d'identité.
5. Le fournisseur d'identité construit ensuite une assertion SAML reprenant le contexte sécurisé de l'authentification. Comme il s'agit d'une liaison POST, l'assertion est signée numériquement et placée dans une réponse SAML, elle-même placée dans un formulaire HTML caché. Ce message est ensuite envoyé au user-agent comme réponse à son authentification.
6. Le navigateur renvoie ensuite une requête POST vers le fournisseur de services avec la réponse signée (soit à l'aide d'une action effectuée par l'utilisateur ou par l'intermédiaire de l'exécution d'un script). Le fournisseur reçoit la réponse et vérifie d'abord la signature de l'assertion SAML. Il traitera ensuite le contenu de l'assertion afin de créer un contexte sécurisé d'authentification localement. Il localise ensuite la ressource et renvoie une redirection vers la ressource demandée initialement.
7. Une vérification d'accès est effectuée afin d'établir si l'utilisateur a l'autorisation adéquate d'accéder à la ressource. La ressource est renvoyée si l'autorisation est valide.

Le flux initié par le fournisseur d'identité est illustré à la Figure 7-6 et est décrit à la suite de celui-ci.

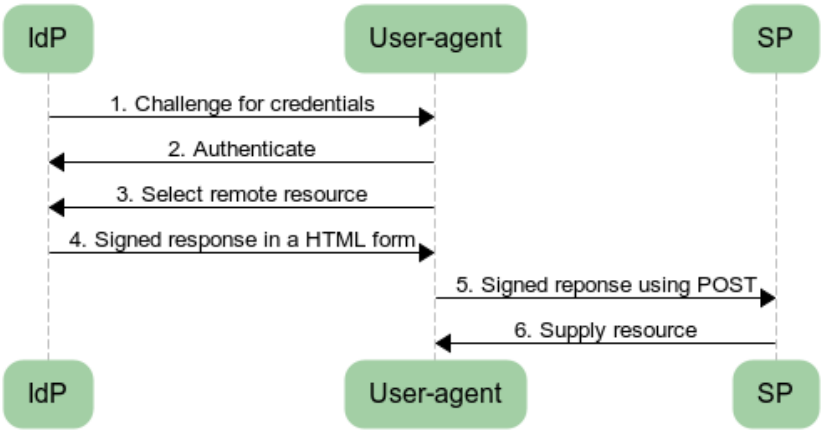


Figure 7-6 : Flux SAML 2.0 initié par le fournisseur d'identité avec liaison POST

1. Si le fournisseur d'identité n'a pas en sa possession un contexte sécurisé d'authentification pour l'utilisateur, ce dernier se verra demander de fournir ses identifiants au fournisseur d'identité.
2. L'utilisateur fournit des identifiants valides et un contexte sécurisé d'authentification est créé pour l'utilisateur chez le fournisseur d'identité.
3. L'utilisateur sélectionne une option du menu ou un lien sur le fournisseur d'identité pour demander l'accès au site web du fournisseur de services.
4. Le service d'authentification unique du service d'identité construit une assertion SAML représentant le contexte sécurisé de l'authentification. Comme la liaison POST est utilisée, l'assertion sera signée numériquement avant qu'elle ne soit placée dans un message SAML, lui-même placé dans un formulaire HTML caché. Ce message est ensuite envoyé au navigateur en tant que réponse.
5. Le navigateur renvoie ensuite une requête POST vers le fournisseur de services avec la réponse signée (soit à l'aide d'une action de l'utilisateur ou par l'intermédiaire de l'exécution d'un script). Le fournisseur reçoit la réponse et vérifie d'abord la signature numérique de l'assertion SAML et traitera ensuite le contenu de l'assertion afin de créer un contexte sécurisé d'authentification localement. Il localise ensuite la ressource et renvoie une réponse de redirection vers la ressource demandée initialement.
6. Une vérification d'accès est effectuée afin d'établir si l'utilisateur a l'autorisation adéquate pour accéder à la ressource. La ressource est renvoyée si l'autorisation est valide.

Seuls deux schémas ont été représentés ici mais il y a de nombreuses possibilités que SAML permet de mettre en place. La modularité des spécifications offre, en effet, de nombreux scénarios possibles.

7.4.3 Profil de découverte des fournisseurs d'identité

Ce profil décrit un mécanisme permettant aux fournisseurs de services de découvrir le fournisseur d'identité d'un utilisateur. Ce mécanisme a été conçu pour le profil Web Browser SSO. Il fonctionne sur base d'un cookie commun entre les différents fournisseurs d'identité contenant la liste de ceux-ci. A chaque nouvelle session d'authentification unique, une entrée contenant l'identifiant du fournisseur d'identité est ajoutée à la liste. L'identifiant du fournisseur d'identité ayant établi la dernière session est toujours repris en dernière position de la liste.

La manière de mettre en place ce cookie est laissée à l'appréciation du fournisseur d'identité. Certaines pistes sont cependant mentionnées dans la section correspondante de la spécification.

Lorsqu'un fournisseur de services a besoin de découvrir quel est le fournisseur d'identité utilisé par un utilisateur, il fait appel à un mécanisme d'échange lui présentant le cookie discuté précédemment.

La manière de lire le cookie est spécifique à l'implémentation du fournisseur de services. Les pistes de telles implémentations sont également suggérées dans la spécification du profil.

7.4.4 Profil de déconnexion unique

Pour rappel ce profil définit un mécanisme de déconnexion distribuée initié par le fournisseur d'identité, l'utilisateur ou le fournisseur de services. La Figure 7-7 représente une déconnexion unique initiée par l'utilisateur. Les deux autres cas ne sont pas abordés mais ils peuvent être déduits facilement de celui-ci.

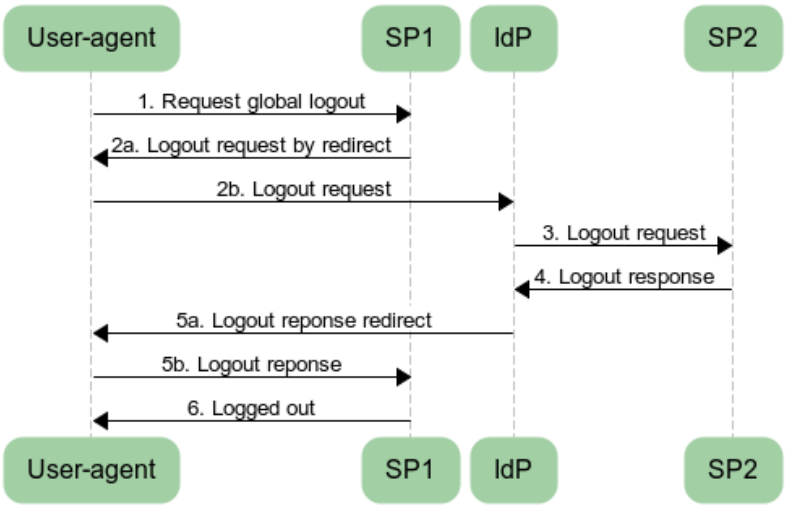


Figure 7-7 : Déconnexion distribuée SAML 2.0

1. Un utilisateur est authentifié sur le fournisseur d'identité et est en train d'interagir avec le fournisseur de services SP1. L'utilisateur décide de mettre fin à sa session et choisit un lien sur le fournisseur de services SP1 afin d'effectuer une déconnexion unique.
2. Le fournisseur de services SP1 détruit la session d'authentification locale de l'utilisateur et envoie au fournisseur d'identité un message de déconnexion. Ce message est signé numériquement et est transmis à l'aide de la liaison de redirection HTTP. Le fournisseur d'identité vérifie que la demande provient bien d'un fournisseur de services connu et dans lequel il a placé sa confiance. Le fournisseur d'identité exécute la requête et détruit toutes les sessions de l'utilisateur.
3. Le fournisseur d'identité détermine qui sont les autres fournisseurs de services participants à la session d'authentification unique et leur envoie un message de déconnexion. Dans ce cas SP2 reçoit un message de déconnexion.
4. Le fournisseur de services SP2 renvoie une réponse, signée numériquement, contenant le statut approprié au fournisseur de services.
5. Le fournisseur d'identité renvoie un message contenant un code de réponse approprié à l'initiateur de la déconnexion (SP1 dans ce cas). Cette réponse est également signée numériquement et est renvoyée, dans ce cas, avec la liaison de redirection HTTP.
6. Finalement, le fournisseur de services SP1 informe l'utilisateur qu'il est déconnecté de l'intégralité des fournisseurs.

7.5 Considérations en termes de sécurité

SAML définit plusieurs mécanismes de sécurité pour se protéger de différentes menaces. Il reste, cependant, basé sur un partenariat de confiance préétabli entre le fournisseur de services et les fournisseurs d'assertions. SAML est, de ce fait, souvent implémenté en intégrant une infrastructure à clés publiques (PKI). C'est d'ailleurs une des recommandations de SAML.

Ces différents mécanismes sont définis dans chacune des liaisons SAML. Un aperçu général de ce qui est recommandé peut être défini comme suit :

- l'utilisation de SSL 3.0 ou TLS 1.0 est recommandée, là où l'intégrité des messages et la confidentialité des messages sont requis ;
- lorsqu'un acteur demande une assertion à un fournisseur d'assertions, une authentification bilatérale est nécessaire et l'utilisation des couches SSL 3.0 et TLS 1.0 utilisant l'authentification mutuelle où l'authentification via des signatures digitales est recommandée ;
- lorsqu'une réponse contenant une assertion est délivrée à un fournisseur de services via le navigateur web, il est nécessaire de signer numériquement la réponse en utilisant la signature XML.

SAML est sujet à différents types d'attaque et principalement toutes celles reprises dans l'état de l'art. La majorité des considérations en termes de sécurité peut être consultée dans un document [Hirsch et al., 2005] créé par OASIS et mentionnant chacune des attaques auxquels il est sujet suivant la spécification implémentée et les mesures à prendre contre ce type d'attaque.

Notons tout de même que parmi toutes ces attaques, le risque principal pour SAML sont les attaques par déni de service. En effet, le traitement des requêtes SAML peut être relativement lourd et certainement lorsqu'il y a chiffrement et signature. Pour cette raison le pirate n'aura pas de mal à rendre le service indisponible puisqu'il lui suffira de générer un grand nombre de requêtes simultanément. Les identifiants éphémères peuvent contribuer à limiter ce type d'attaque.

7.6 Respect de la vie privée

SAML est souvent déployé dans des scénarios où les exigences suivantes en matière de vie privée doivent être prises en compte :

- la capacité d'un utilisateur à contrôler comment leurs données d'identité sont utilisées et partagées ;
- aux mécanismes qui inhibent leurs actions, sur plusieurs fournisseurs de services, s'ils ne sont pas corrélés de façon adéquate.

Les mécanismes qui permettent d'offrir cette protection sont les suivants :

- SAML supporte l'établissement de pseudonymes établis entre un fournisseur d'identité et un (voir plusieurs) fournisseur(s) de services ;
- SAML supporte les identifiants à utilisation unique ou éphémère. De tels identifiants nous assurent que, lors de chaque accès d'un utilisateur à un fournisseur de services par l'intermédiaire d'une authentification unique, les fournisseurs de services seront incapables de les reconnaître en tant qu'individu ;
- les mécanismes de contexte d'authentification permettent aux utilisateurs de s'authentifier à un niveau d'assurance suffisant et adapté aux ressources auxquelles il tente d'accéder ;
- SAML permet d'exprimer, entres différents fournisseurs de services, le fait qu'un utilisateur approuve certaines opérations entre ceux-ci.

L'identité fédérée est également respectueuse de la vie privée. Ceci est rendu possible par la création d'identifiant éphémère. Ce mécanisme garantit l'anonymat avec l'identifiant d'un autre système.
Le seul risque auquel l'aspect vie privée de SAML est exposé est la répétition d'actions précises par une personne. Ces actions, répétées pour un même domaine, ne sont pas respectueuses de la vie privée mais ne dépendent pas de SAML.

7.7 Synthèse

SAML est le standard par excellence pour l'identité fédérée. Il permet, par ce biais, une ouverture entre le monde de l'entreprise et les domaines autonomes. Ce standard est une clé pour la convergence de la gestion d'identité fédérée et c'est de cela qu'il tire sa popularité et son implémentation en masse. La sécurité et la richesse des messages qu'il véhicule sont également deux de ses atouts majeurs.

SAML est également un standard souple permettant de définir très facilement de nouveaux types de messages, un type précis de communication à utiliser, d'ajouter de nouvelles fonctionnalités,… A partir du moment où un partenariat avec une entité de confiance peut être mis en place, SAML est, sans nul doute, le protocole adéquat pour couvrir un grand nombre de cas d'utilisation différents.

La modularité de SAML à un prix, elle nécessite un travail d'implémentation relativement conséquent afin de couvrir différents scénarios de façon appropriée. C'est sans doute la raison pour laquelle il est le candidat principal pour les grandes entreprises, les institutions, les universités,… SmalS[1], société ICT fournissant des services informatiques à certaines institutions de l'Etat (dont Fedict), est d'ailleurs un utilisateur intensif de SAML. Par le biais de l'utilisation de ce protocole entre les différentes institutions et par son aspect modulaire, il les rend facilement interopérables les unes avec les autres. Il laisse en effet la possibilité, aux personnes qui l'implémentent, de choisir ce dont ils ont besoin au sein des différentes couches du protocole.

Notons également que les artefacts, peu abordés dans l'étude de ce protocole, permettent d'améliorer les communications à bande limitée telles que les réseaux cellulaires et autres…

Bien que les recommandations en termes de sécurité soient appropriées, le large choix offert au fournisseur de services et d'identité, peut laisser apparaître certaines failles de sécurité. Il est donc primordial de consulter et de se fier aux recommandations en termes de sécurité et vie privée disponibles dans le document de même nom. Ceci est applicable même si la modularité de SAML aurait facilement tendance à nous en décoller une fois les différents principes de ce protocole adoptés.

[1] https://www.smals.be/

8 Grilles de comparaison

Ce chapitre est une analyse comparative, sous différents angles, des systèmes étudiés tout au long de ce mémoire.

Comme vous avez pu le constater dans les chapitres précédents, certains systèmes d'authentification unique sont appropriés pour des cas d'utilisation particuliers, d'autres sont plus sécurisés mais plus difficile à mettre en place, certains respectueux de la vie privée d'autres pas,… L'offre est très variée et l'utilisateur est souvent quelque peu démuni lors du choix d'un système approprié à ses besoins.

Cette section a pour objectif de mettre à disposition de l'utilisateur un outil pour l'aider à choisir le système adéquat en fonction de ses désidératas et de ses exigences. Pour ce faire un tableau global sera réalisé avec les différents aspects de chacun des protocoles étudiés. Chacun de ces aspects sera analysé et décrit en détail. Le tableau reprenant l'aperçu global conduira l'utilisateur dans le choix d'un système suivant un cas particulier et le contexte de l'implémentation d'un système d'authentification unique. A la fin de ce chapitre est expliqué avec quelques exemples comment le tableau récapitulatif peut être utilisé à bon escient.

8.1 Préambule

Les données reprises dans ces analyses n'ont pas la prétention d'être les outils idéaux pour le choix d'un système. Ils devront néanmoins, avec l'approche faite tout au long de ce mémoire, vous donner une idée claire de ce qui est possible et envisageable avec chacun des modules d'authentification unique étudiés ici.

Chaque ligne/colonne du tableau contient d'une part chacun des systèmes concernés et d'autre part les critères mesurés. Une explication, reprise en gras, en tête de tableau expliquera, si cela est nécessaire, les sous-mesures que cela englobe et donc une définition de ce que ce terme signifie dans le cadre de cette étude.

Les éléments de ce tableau seront complétés soit :

- par une échelle de 0 à 3 points où figure :
  - le symbole «  » ou «  » si le critère étudié représente un avantage pour le protocole ;
  - le symbole «  » ou «  » si le critère étudié représente un inconvénient pour le protocole ;
- par un symbole «  » représentant la présence d'une fonctionnalité, un symbole «  » s'il s'agit d'une partie de la fonctionnalité ou par un symbole «  » signifiant l'absence de la fonctionnalité ;
- par des informations textuelles ;
- par un champ noté « **N/A** » (Not Applicable) pour mentionner que le critère n'est pas applicable pour le protocole étudié.

Lorsque des moyennes ont été calculées, l'échelle de 0 à 3 passe à une échelle de 0 à 6 par l'utilisation de demi-points (représenté par la première moitié du symbole correspondant) afin de rester suffisamment précis dans les données. Les points obtenus lors de moyenne ont été arrondis à l'unité (ou la demi-unité) supérieure.

En règle générale et pour le calcul de la moyenne, un symbole complet représente 1 point et la moitié de ce symbole un demi-point. Une valeur « **N/A** » n'est pas prise en compte dans le calcul de la moyenne.

En cas de besoin et au cas par cas, la manière de coter sera précisée avant ledit tableau.

8.2 Tableaux comparatifs

8.2.1 Qualité de la documentation

La Table 1 s'attarde principalement sur l'information contenue dans les différentes spécifications, leur type et les documents annexes pour chacun des protocoles étudiés.

Le premier bloc du tableau indique dans quel état d'avancement se trouve la spécification. De manière générale, la spécification passe d'abord par un état de **brouillon** (valide pendant 6 mois) pour devenir une **norme** (si agréée par un organisme de normalisation autrement il s'agit d'une publication) et enfin un **standard** (norme utilisée en masse).

La documentation peut également être une **implémentation** lorsqu'il s'agit de la mise en œuvre d'une spécification. Dans ce cas la spécification utilisée sera reprise entre parenthèses dans la même case.

Le bloc suivant reprend le mois et l'année de la **dernière mise à jour** des spécifications du protocole et si ces documents ont encore des perspectives d'évolution. On entend par « **perspectives d'évolution** » des avancements publiés (extensions, mises à jour, errata,…) ou une communauté se penchant encore sur le protocole et donc sur une éventuelle nouvelle version de sa spécification. On mentionnera également l'organisme ayant **publié** la (ou les) différente(s) spécification(s) composant le protocole.

L'avant-dernier bloc porte sur les différents critères suivants concernant la documentation:

- la **clarté** :
    - le document contient des illustrations, des exemples de requête ;
    - tous les termes sont clairs lors d'une première lecture ;
    - tous les éléments sont décrits en détail, ne laissant place à aucune confusion ;
- la **complétion** :
    - la documentation possède un ensemble de documents annexes tels qu'un glossaire, un aperçu,… ;
    - la documentation possède des documents explicatifs complémentaires tels qu'une approche chef de projet, des considérations en termes de vie privée et de sécurité,… ;
    - la spécification est suffisamment détaillée ;
- l'**organisation** :
    - si une documentation est modulaire à souhait (et dans ce cas, elle contient un aperçu global clair sur les différents modules);
    - si l'agencement au sein des documents est approprié ;
- l'**accessibilité** étudie si la documentation est facile d'accès, si son référencement au sein d'un moteur de recherche tel que Google est approprié.

Le dernier bloc concerne la **qualité de la documentation**. Il s'agit d'une note globale donnée à la documentation vis-à-vis des quatre derniers critères quantifiables. Il s'agit donc d'une moyenne des 4 critères précédents.

Table 1 : Qualité de la documentation

	OAuth 1.0a	OAuth 2.0	Facebook Connect	OpenID 2.0	OpenID Connect 1.0	SAML 2.0
Brouillon		✓			✓	
Norme	✓			✓		
Standard						✓
Implémentat-ion			✓ (OAuth 2.0)			
Perspective d'évolution		✓	✓		✓	✓
Date de la dernière MAJ	Avril 2010	Mai 2012	Mai 2012	Décembre 2007	Avril 2012	Mars 2005
Publié par	Communauté OAuth	Communauté OAuth	Facebook	Communauté OpenID	Communauté OpenID	OASIS
Clarté	⊕	⊕⊕	⊕⊕⊕	⊕⊕	⊕⊕⊕	⊕⊕⊕
Complétion		⊕⊕	⊕⊕	⊕	⊕⊕	⊕⊕⊕
Organisation	⊕	⊕⊕	⊕⊕	⊕⊕	⊕⊕⊕	⊕⊕
Accessibilité	⊕⊕⊕	⊕⊕		⊕⊕⊕	⊕⊕⊕	⊕
Moyenne	1,25	2	1,75	2	2,75	2,25
Qualité de la documentation	⊕◐	⊕⊕	⊕⊕	⊕⊕	⊕⊕⊕	⊕⊕◐

OpenID Connect se démarque des autres protocoles en termes de qualité de la documentation. La raison est très simple : les spécifications sont modulaires à souhait, les documents et l'aperçu général sont clairs, les différentes spécifications sont truffées d'exemples. Bref, OpenID Connect fait office de référence en la matière. C'est d'autant plus à son avantage que la communauté OpenID n'en est encore qu'a la version brouillon de ces spécifications et qu'il est donc possible que son manque relatif à la complétion soit comblé dans le futur.

En comparaison avec SAML qui est la référence en matière de complétion, il n'y a pas de document, au sein d'OpenID Connect, abordant une approche de chef de projet, un aperçu technique,… Pour terminer, OpenID Connect n'a pas encore de version publiée et pourtant son référencement est déjà adéquat.

Il n'y a donc pas de document parfait mais certains protocoles s'approchent tout de même d'une documentation adéquate et appropriée sur tous les points de vue. OpenID Connect en est l'exemple. En contrepartie, OAuth 1.0a possède une documentation relativement maigre et très limitée en comparaison avec ses concurrents. Un seul flux y est décrit et celui-ci laisse place à beaucoup d'interprétations quant aux possibles implémentations qui peuvent en être faites. Ce point a été amélioré dans la version 2 du protocole.

8.2.2 Richesse des données

La Table 2 aborde la richesse des données pouvant être obtenues par un fournisseur de services de la part d'un fournisseur d'identité. Il s'agit d'un des principaux critères de sélection pour un fournisseur de services. En effet ces données peuvent être utilisées à diverses fins :

- améliorer l'expérience utilisateur ;
- pouvoir cibler la publicité présentée à l'utilisateur ;
- …

Cette richesse de données est également intéressante pour un fournisseur d'identité. Il va en effet pouvoir s'en servir pour appâter les potentiels fournisseurs de services afin d'accroitre sa popularité et peut-être se démarquer des autres modules permettant de faire de l'authentification unique.

Les utilisateurs et les sociétés un tant soit peu sensibilisés par l'aspect vie privée de leurs données, pourraient justement tenter de fuir ce type de système. Alors qu'une majorité d'utilisateurs de la toile l'utiliseront sans trop réfléchir.

Le tableau est composé des deux parties suivantes :

- les **données de profil**: indication sur la quantité de données qu'il est possible d'obtenir au sujet d'un utilisateur (moyennant des autorisations quand elles sont d'application) ;
- les **moyens utilisés** : représente les moyens utilisés par le fournisseur de services pour obtenir l'information concernant l'utilisateur. Il s'agit des techniques utilisées, ou des outils, API's mis à disposition du fournisseur de services par le fournisseur d'identité et permettant d'obtenir les informations souhaitées concernant l'utilisateur.

Table 2 : Richesse des données

	OAuth 1.0a	OAuth 2.0	Facebook Connect	OpenID 2.0	OpenID Connect 1.0	SAML 2.0
Données de profil	⊕⊕	⊕⊕	⊕⊕⊕	⊕⊕	⊕⊕	⊕⊕⊕
Moyens utilisés	Transmis dans la variable scope	Transmis dans la variable scope	Transmis dans la variable scope + API Graph, API REST, FQL[1], XFBML[2], canvas pages[3]	Via la spéc. Attribute Exchange 1.0	Requête sur le endpoint UserInfo	Génération d'assertions d'attribut

Globalement, la diversité des données de profil pouvant être transmises par chacun des systèmes étudiés est, à peu de chose près, identique. Cependant elle est difficilement quantifiable sans analyser et comparer plusieurs implémentations entre elles. Facebook Connect et SAML obtiennent la cote maximale vis-à-vis de leurs concurrents. La raison est simple SAML permet de définir un format de message contrairement à ses opposants dont le format est moins souple. Cela lui permet notamment de pouvoir transmettre des assertions concernant tout et n'importe quoi. Facebook Connect, quant à lui, est un leader en termes de données d'utilisateur. En effet, les données qu'il possède sont nombreuses et vont au delà de ce à quoi l'on peut s'attendre avec OAuth 2.0 et les autres protocoles. Non seulement il s'impose comme un des grands fournisseurs d'identité sur le marché mais il met également à disposition plusieurs API's et méthodes afin d'obtenir des informations sur l'utilisateur.

Les moyens utilisés pour obtenir toutes ces informations sont par défaut limités à la seule méthode mentionnée dans la spécification et reprise dans la table. Il est évidemment possible d'en mettre d'autre en place lors de l'implémentation des spécifications. Facebook Connect en est l'exemple clé.

8.2.3 Fonctionnalités et services

La Table 3 fait le listing des fonctionnalités et services disponibles pour chacun des protocoles. On indiquera simplement dans chacune des cases correspondantes à la fonctionnalité si elle est disponible pour le système mentionné.

[1] flux coté client
[2 & 3] flux coté serveur

S'il existe une pseudo équivalence pour une implémentation à l'aide d'autres services, si une partie de la fonctionnalité est décrite dans la spécification ou si la fonctionnalité dépend de l'implémentation du fournisseur d'identité, les cases seront complétées par le symbole « » représentant un demi-point. A titre d'exemple de pseudo équivalence, la déconnexion unique au sein d'OpenID dépend de l'implémentation par son fournisseur d'identité.

Si la fonctionnalité ne peut être déployée, si elle est fortement déconseillée ou non couverte par le protocole à proprement parler, une croix rouge sera indiquée. Les différents services et fonctionnalités ne seront pas décrits en détail puisqu'ils ont été vus tout au long du mémoire.

La moyenne par protocole des différents points obtenus pour chacun des services a été calculée en fin de tableau sous le champ **fonctionnalités et services**. Cette moyenne a été multipliée par 3 afin de rester sur la même échelle que celle utilisée par les autres critères de l'aperçu global repris à la Table 10.

Table 3 : Fonctionnalités et services

	OAuth 1.0a	OAuth 2.0	Facebook Connect	OpenID 2.0	OpenID Connect 1.0	SAML 2.0
Authentification unique	✓	✓	✓	✓	✓	✓
Fédération d'identité	✗	✗	◔	◔	◔	✓
Délégation de l'autorisation	✓	✓	✓	◔	✓	◔
Déconnexion unique	✗	✗	✓	◔	✓	✓
Service de découverte	✗	✗	✗	✓	✓	✓
Enregistrenement de nouveau client	✗	✗	✗	✓	✓	✓
Multi profils	✗	✗	✗	◔	◔	◔
Autres fonctionnalités	✗	✗	✗	✗	✗	✓
Moyenne	0,75	0,75	1,31	1,88	2,25	2,63
Fonctionnalités et services	✓	✓	✓◔	✓✓	✓✓◔	✓✓✓

Il s'agit pour tous les systèmes repris dans la Table 3, de module d'authentification unique au centre de l'étude de ce mémoire.

La fédération d'identité n'est pas décrite à proprement parler au sein de toutes les spécifications. Le standard SAML 2.0 la définit clairement et propose tous les mécanismes pour la mettre en œuvre. Facebook Connect, OpenID et OpenID Connect, quant à eux, permettent également de faire de la fédération d'identité en une certaine mesure et/ou moyennant le choix d'un fournisseur d'identité approprié. OpenID Connect améliore même l'approche de la fédération d'identité par l'intermédiaire de sa spécification OpenID Session Management.

Concernant la délégation d'autorisation, tous les modules en sont dotés d'une manière ou d'une autre. Rappelons tout de même que OAuth 1.0a / 2.0 et OpenIDConnect restent les pionniers en la matière.

La déconnexion unique n'est pas décrite dans les versions d'OAuth. Facebook Connect l'a cependant mise en place dans son implémentation. Rappelons que OAuth est principalement un mécanisme de délégation d'autorisation. OpenID 2.0 ne la décrit que

très peu dans sa spécification, il revient donc au fournisseur d'OpenID de le mettre en place et cela dépendra donc de celui-ci.

Le service de découverte a été typiquement initié par OpenID, pionnier en la matière. Cet aspect n'est pas repris dans les spécifications OAuth et est indisponible dans Facebook Connect car il n'est pas d'application.

L'enregistrement de nouveau client est uniquement possible via les protocoles OpenID 2.0, OpenID Connect 1.0 et SAML 2.0.

La fonctionnalité multi profils, pour rappel, permet à un utilisateur de sélectionner un profil contenant certaines données pouvant être transmises au fournisseur de services et variant suivant le profil. Ce service est généralement disponible dans les implémentations d'OpenID 2.0, on s'imagine aisément OpenID Connect 1.0 le faire. SAML permet cette fonctionnalité mais ne la décrit pas dans les spécifications étudiées. Facebook Connect ne le permet pas. En effet, les autorisations font office de filtres sur les données transférées et ne permettent donc pas de définir des données différentes pour un même attribut.

Pour résumer OpenID Connect 1.0 et SAML 2.0 décrivent une multitude de fonctionnalités et services (comme on peut le constater). Pour rappel, SAML permet de définir un format de messages, ouvrant la porte à toute une multitude de services au delà de ceux décrits dans la Table 3. C'est d'ailleurs la raison pour laquelle, SAML est le seul protocole à proposer d'autres fonctionnalités.

8.2.4 Scénarios couverts

La Table 4 représente la richesse d'un protocole vis-à-vis des scénarios qu'il couvre. En effet, pour ne mentionner que les deux extrêmes, OAuth 1.0 ne couvre qu'un seul scénario alors que OpenID Connect et SAML en décrivent une multitude. Bien que OAuth puisse être déployé dans d'autres situations, ces autres cas ne sont nullement décrits au sein de la spécification laissant son lecteur indécis quant à la bonne façon d'implémenter le protocole pour un flux sortant du traditionnel décrit. SAML, en contrepartie décrit une multitude de scénarios (profils), protocoles et autres laissant à l'utilisateur le choix d'implémenter son flux comme il le souhaite avec les recommandations adéquates en termes d'implémentation.

Le premier critère mentionne, à titre indicatif et de synthèse, le **nom des différents flux** possibles au sein de chacun des protocoles.

Le second critère **flux d'extension** aborde la possibilité de définir un flux de manière un peu plus ouverte permettant aux personnes qui s'occupent de la mise en place du protocole de définir certains critères/variables tout en ayant un guide permettant de le faire. Si ce flux porte un nom il sera mentionné dans le même cadre entre parenthèses.

Le critère **fournisseur d'identité externe** aborde la possibilité de coupler le fournisseur de services à un fournisseur d'identité externe quelconque.

Le dernier critère de ce tableau porte **les scénarios couverts** par les différentes spécifications d'application. Cette richesse est donc mesurée sur base des trois critères précédents, le poids des critères ayant été réparti comme suit :

- le nombre de flux repris dans le critère **nom des flux** → 1,5 unité (à raison de 0,25 unité par flux) ;
- la présence d'un flux d'extension → 1 unité ;
- le couplage possible d'un module avec un fournisseur d'identité externe → 0,5 unité.

Table 4 : Scénarios couverts

	OAuth 1.0a	OAuth 2.0	Facebook Connect	OpenID 2.0	OpenID Connect 1.0	SAML 2.0
Nom des flux	Flux unique	- code d'autorisation; - octroi implicite; - identifiants par mot de passe du propriétaire de la ressource ; - identifiants du client.	- coté client (octroi implicite) - coté serveur (code d'autorisation)	Flux unique (avec 2 modes de communication)	- Octroi implicite - Code d'autorisation - Application native pour un quatrième intervenant	Nombreux flux possible décrit au sein des différentes spécfications
Flux d'extension	✗	✓ (flux étendu)	✗	✗	✓[1]	✓
Fournisseur d'identité externe	✓	✓	✗	✓	✓	◔
Moyenne	0,75	2,5	0,5	1	3	2,75
Scénarios couverts	✓	✓✓◔	◔	✓	✓✓✓	✓✓✓

OpenID Connect, OAuth 2.0 et SAML 2.0 obtiennent un score élevé d'un point de vue de la richesse des scénarios. En effet, ces protocoles décrivent différents flux, prennent en compte les fournisseurs d'identité externes et/ou offrent un flux étendu. Ce tableau ne le mentionne pas mais OpenID Connect définit sa spécification en termes de endpoints. Les endpoints peuvent donc être déployés où bon nous semble. De ce fait, OpenID Connect obtient le maximum de points pour le nombre de flux.

SAML se voit sanctionner, dans ce tableau, par son aspect limité au cercle de confiance. Notons qu'il est pourtant bien plus riche qu'OpenID Connect en termes flux. Il est possible de le mettre en place pour un fournisseur d'identité externe, il nécessite cependant un effort considérable pour y parvenir raison pour laquelle il a reçu un demi point pour ce critère. Autrement, pour tous les autres cas, SAML 2.0 reste la référence en la matière. Les spécifications assertions, protocoles, et liaisons permettent d'imaginer une solution pour pratiquement tous les scénarios possibles.

8.2.5 Contre-mesures face aux menaces

La Table 5 évalue les contre-mesures décrites au sein de chacune des spécifications vis-à-vis des menaces citées dans l'état de l'art.

L'étude a été réalisée à l'aide des spécifications des protocoles (ou documents annexes) et des considérations en termes de sécurité reprises dans les différents chapitres de ce mémoire. Lorsque différents flux étaient décrits au sein d'un protocole, la sécurité a été étudiée sur base du flux le plus sécurisé. L'approche inverse (habituellement utilisée dans des approches liées à la sécurité) aurait en effet donné une image peu représentative de certains protocoles.

L'attribution des points aux systèmes vis-à-vis de chacune des attaques a été réalisée sur base des points suivants:

- la sensibilisation de l'utilisateur au type d'attaque ;

[1] A l'aide des spécifications OAuth 2.0 Multiple Response Type Encoding Practices et Messages

- l'adéquation des mots-clés[1] utilisés dans les différents documents ;
- la présence d'un mécanisme de protection / d'une méthode adéquate pour se protéger de l'attaque ou la preuve de l'inefficacité d'une menace vis-à-vis du protocole ;
- l'efficacité du mécanisme mis en place.

Pour résumé, un point est attribué par réponse positive aux questions suivantes :
- Est-ce que l'utilisateur est averti du risque de façon claire ?
- Est-ce qu'un mécanisme ou un conseil adéquat est donné pour se protéger du type d'attaque ?
- Est-ce que le mécanisme est efficace ?

En règle générale, le protocole n'obtient pas de point si aucune sensibilisation au sujet de l'attaque n'est présente au sein des documents concernés ou que les recommandations peuvent être démontrées comme inefficaces. A l'inverse, trois symboles seront repris si l'utilisateur est sensibilisé correctement à la problématique et que le type d'attaque ne représente pas de menace pour le protocole étudié grâce à l'efficacité du mécanisme mis en place au sein du protocole et aux recommandations adéquates en termes de sécurité.

En ce qui concerne l'implémentation d'un protocole, le symbole suivant «  » sera utilisé pour mentionner à quel type d'attaque on s'expose pour l'implémentation de ce protocole. Une case vide dans le cadre d'une implémentation signifie qu'il n'y a aucun risque. Le nombre de symboles «  » représente le degré auquel le système s'expose vis-à-vis de la menace citée.

Le degré d'exposition aux menaces des systèmes n'a pas été mesuré car il est fort semblable d'un protocole à un autre. En effet, la liste des attaques n'est pas une liste exhaustive de toutes les attaques existantes mais uniquement de celles s'appliquant dans le cadre des méthodes étudiées dans ce mémoire. Cependant, notons tout de même que les systèmes d'authentification unique possédant un service de découverte sont plus souvent sujet à des attaques visant le DNS. Les attaques par force brute, le déni de service ou encore l'hameçonnage feront l'objet d'une attention particulière pour ces systèmes.

Une **moyenne** par protocole sur la sécurité dans son ensemble a été calculée sur base des points obtenus pour chacune des attaques. Elle est ensuite représentée sous le critère **contre-mesures face aux menaces** par le nombre de symboles correspondants.

Afin de remettre Facebook Connect dans la même philosophie de calcul de moyenne que les autres protocoles, les symboles «  » ont été remplacés par des symboles «  » en prenant le complémentaire. Une attaque ayant obtenu un résultat de 2 symboles «  » s'est donc vue attribuer un symbole «  ». Rappelons que les cotes « **N/A** » n'entrent pas en ligne de compte lors du calcul de la moyenne.

[1] Au sein des différents documents portant sur la sécurité, certains mots-clés tels que « requis », « facultatif », « doit », « devrait »,… sont utilisés afin de décrire les recommandations en termes de sécurité. L'adéquation de cette notion a également été prise en compte. A titre d'exemple, un système conseillant l'utilisation de TLS ou SSL est donc plus exposé aux menaces qu'un système l'exigeant.

Table 5 : Contre-mesures face aux menaces

	OAuth 1.0a	OAuth 2.0	Facebook Connect	OpenID 2.0	OpenID Connect 1.0	SAML 2.0
Attaque par cryptanalyse	✓	✓ ✓	N/A	✓ ✓	✓ ✓ ✓	✓ ✓ ✓
Attaque CSRF	✓	✓ ✓ ✓	✗ ✗ ✗	✓ ✓ ✓	✓ ✓ ✓	N/A
Attaque par fixation de session	✓ ✓	✓ ✓ ✓	N/A	✓ ✓	✓ ✓ ✓	✓ ✓
Attaque par force brute	✓ ✓	✓ ✓	N/A		✓ ✓	N/A
Attaque par rejeu	✓	✓ ✓ ✓	✗ ✗ ✗	✓ ✓ ✓	✓ ✓ ✓	✓ ✓ ✓
Attaque de l'homme du milieu	✓	✓ ✓	N/A	✓ ✓	✓ ✓ ✓	✓ ✓ ✓
Clickjacking	✓ ✓	✓ ✓ ✓	✗ ✗	✓ ✓	✓ ✓ ✓	N/A
Déni de service	✓ ✓	✓ ✓	N/A	✓ ✓ ✓	✓ ✓	✓ ✓
Ecoute du réseau	✓	✓ ✓ ✓	✗ ✗ ✗	✓ ✓ ✓	✓ ✓ ✓	✓ ✓ ✓
Hameçonnage	✓	✓ ✓ ✓	✗	✓ ✓ ✓	✓ ✓ ✓	✓ ✓ ✓
Usurpation d'identité	✓	✓ ✓ ✓	✗ ✗ ✗	✓ ✓	✓ ✓ ✓	✓ ✓ ✓
Moyenne	1,36	2,63	0,5	2,27	2,81	2,75
Contre-mesures face aux menaces	✓	✓ ✓	✓	✓ ✓	✓ ✓ ✓	✓ ✓ ✓

OAuth 1.0a est un mauvais élève en matière de contre-mesures face aux menaces : sa spécification est en effet beaucoup trop laxiste en matière de sécurité. Il sensibilise le fournisseur de services et le fournisseur d'identité sur la majorité des différentes attaques mais ne fournit pas beaucoup de conseils sur comment contourner ces problèmes. De plus, les mécanismes de sécurité tels que TLS, SSL sont conseillés alors qu'ils devraient être obligatoires.

OAuth 2.0 est nettement plus sécurisé, les recommandations sont nombreuses et claires. Les mécanismes conseillés sont souvent adéquats mais ne possèdent pas toujours l'efficacité escomptée. La sensibilisation aux aspects de la cryptanalyse (les mécanismes de chiffrement sont cependant résistants), de l'attaque par force brute, du déni de service et de l'attaque de l'homme du milieu n'est pas complète ou expliquée de façon peu claire.

Facebook Connect compte énormément sur l'utilisation d'un navigateur sécurisé reportant ainsi la sécurité sur le poste client. Ce point a été pris en compte dans tous les critères mesurés. Rappelons qu'une analyse de la sécurité de Facebook Connect a été faite dans le chapitre correspondant, apportant une majorité de réponses aux notes attribuées à ce protocole. Notons que la sensibilité des informations que possède Facebook au sujet de l'utilisateur et au vu de la section respect vie privée, l'attaque par usurpation d'identité écope de la plus mauvaise cote. L'attaque par force brute sur Facebook a été testée manuellement mais n'est pas suffisamment pertinente que pour en tirer une conclusion. Après trois tentatives, l'utilisateur se voit proposer la possibilité de changer son mot de passe mais l'authentification est toujours possible et aucune désactivation temporaire n'a été remarquée. Facebook Connect n'est certainement pas l'exemple idéal en termes d'implémentation de OAuth 2.0. Il s'expose en effet à des failles de sécurité importantes.

OpenID, au vu des fonctionnalités qu'il propose, est évidemment plus exposé aux menaces de sécurité, mais il s'en sort plutôt bien. OpenID ne fait cependant aucune sensibilisation contre les attaques par force brute envers les fournisseurs d'identité souhaitant implémenter OpenID. Le clickjacking n'est également pas discuté. Des

protections, contre les navigateurs infectés, sont mentionnées au sein de la spécification rendant OpenID résistant à ce type d'attaque.

OpenID Connect apporte quelques éclaircissements concernant deux types d'attaque. Des mécanismes divers permettent en outre de sécuriser les requêtes de bout en bout. Nous regrettons simplement que ce protocole ne s'étende pas plus sur les sujets d'attaque par force brute, les attaques par fixation de session ainsi que le clickjacking.

Il est difficile de mettre SAML sur le même pied d'égalité que les autres systèmes étant donné que le type de public auquel il est destiné est quelque peu différent. Les attaques dont la source est principalement le « Tout public », se sont vues attribuer une cote « N/A » (afin que cette cote ne soit pas prise en compte dans le tableau récapitulatif). En effet, bien que ce module pourrait être implémenté dans un environnement moins sécurisé, la spécification ne le conseille pas. Rappelons également que SAML recommande l'utilisation d'une infrastructure à clés publiques.

En conclusion, SAML et OpenID Connect 1.0 semblent être les deux protocoles les plus sûrs en matière de sécurité. Ces résultats doivent être relativisés et certainement concernant SAML 2.0 puisque le type de public de ce protocole est différent et que l'aspect sécurité pour l'implémentation hors cercle de confiance n'a pas été étudié. Par ailleurs l'étude a été faite suivant certains types d'attaque que certains auraient peut-être eu tendance à regrouper ou au contraire à détailler. Il s'agit donc ici d'une indication. Ce tableau ne doit pas être pris pour acquis et une étude personnalisée d'un point de vue de la sécurité devrait être faite pour chacun des modules suivant le type d'implémentation et la sensibilité des données qu'elle traite. D'autre part, il y a certainement d'autres failles de sécurité qui pourraient venir s'ajouter à cette liste. Comme c'est déjà le cas pour certaines spécifications récentes reprenant des points d'attention pour des attaques qui n'étaient pas ou peu connues à l'époque de la création des spécifications plus anciennes.

On constatera que des efforts importants de documentation ont souvent été entrepris pour les failles auxquelles les protocoles sont le plus exposés.

8.2.6 Respect de la vie privée

La Table 6 est une analyse du respect de la vie privée. Elle a pour objectif de sensibiliser les personnes souhaitant implémenter un module respectueux de la vie privée envers les utilisateurs (contrairement à la Table 2 qui se focalise sur les données transmises aux fournisseurs de services). Ce type d'analyse devrait être effectué principalement sur les implémentations afin de voir si les recommandations en termes de vie privée ont bel et bien été respectées. L'analyse se basera donc principalement sur les sensibilisations au respect de la vie privée présentes au sein de chacune des spécifications.

Le tableau est composé des critères suivants :

- le **type de public** :
    - tout public : système principalement destiné à toute personne présente sur la toile et s'adressant à un large panel ;
    - indéfini : il est difficile de déterminer un type de public sur base de la spécification. L'étude de l'implémentation du protocole est déterminante pour ce critère ;
    - cercle de confiance : destiné à des cercles fermés ou la relation entre le fournisseur d'identité et le fournisseur de services est étroite (universités, institutions,…) ;

  Il est évident que les recommandations, en termes de vie privée et de sécurité, doivent être plus strictes dans le cas du tout public ;
- **données exposées publiquement** : indication sur la quantité de données qui est exposée au public. Il s'agit des informations concernant l'utilisateur qui

peuvent être exposées. Elles sont parfois nécessaires pour le fonctionnement du protocole mais sont lisibles par le fournisseur de services sans avoir nécessairement établi d'authentification ni reçu les autorisations nécessaires ;

- **données accessibles par le fournisseur de services** : copie exacte du critère **données de profil** de la Table 2. Il est repris dans ce tableau afin d'avoir une vue globale des données concernant la vie privée et voir quelles sont celles qui sont d'application pour une implémentation comme Facebook Connect ;
- **sensibilisation au respect de la vie privée :** ce critère mentionne si une section sensibilisant au respect de la vie privée est présente soit au sein des spécifications elles-mêmes (les mécanismes à mettre en place pour la préserver) soit dans les implémentations, en mentionnant les risques liés à l'utilisation des données de l'utilisateur et leur divulgation ;
- **contrôle par autorisation** : ce critère mentionne si le système d'authentification unique concerné possède bien un module de contrôle des autorisations. Une case vide signifie que cela dépend de l'implémentation par le fournisseur d'identité. Cela a, en effet, un effet négatif sur la sécurité de la vie privée s'il est possible de transmettre d'autres informations que celles mentionnées par défaut sans demander le consentement de l'utilisateur;
- **format des autorisations :** sensibilisation sur la clarté et la compréhension des autorisations que l'utilisateur doit approuver ;
- **respect de la vie privée** : il s'agit d'un critère spécifique aux implémentations. Ce critère mesure l'atteinte à la vie privée indépendamment de la robustesse du système. Il s'agit donc de la sensibilité du système à transmettre des informations concernant la vie privée de l'utilisateur peu importe la personne concernée. Ce critère se base donc essentiellement sur les données exposées publiquement, celles étant exposées au fournisseur de services et le format des autorisations. Un champ « **N/A** » signifie que cela n'est pas mesurable hors implémentation ;
- **risque d'atteinte à la vie privée :** les résultats obtenus concernant **le respect de la vie privée** couplés avec les résultats du critère **contre-mesures face aux menaces** de la Table 5 permettent de définir le critère « risque d'atteinte à la vie privée ».

Table 6 : Respect de la vie privée

	OAuth 1.0a	OAuth 2.0	Facebook Connect	OpenID 2.0	OpenID Connect 1.0	SAML 2.0
Type de public	Indéfini	Indéfini	Tout public	Indéfini	Indéfini	Cercle de confiance
Données exposées publiquement	N/A	Nom d'utilisateur[1]	- nom, image de profil, réseaux, nom d'utilisateur, identifiant - tout ce qui ne dispose pas d'une icone de partage - client_id	URI	- URI / e-mail - client_id	N/A
Données accessibles par le fournisseur de services	⊕⊕	⊕⊕	⊕⊕⊕ - statut courant, ville actuelle, formation, emploi, activité liée aux applications, tranche d'âge[2], paramètres régionaux[2], genre[2] - liste d'amis et l'infos liées (voir Figure 4-6)[2] - toutes les informations nécessitant des autorisations	⊕⊕	⊕⊕	⊕⊕⊕
Sensibilisation au respect de la vie privée			✓	✓	✓	✓
Contrôle par autorisation	✓	✓	✓		✓	
Format des autorisations	N/A	N/A	⊕⊕⊕	N/A	N/A	N/A
Respect de la vie privée	N/A	N/A	⊖⊖	N/A	N/A	N/A
Risque d'atteinte à la vie privée	N/A	N/A	⊖⊖⊖	N/A	N/A	N/A

Difficile de déterminer le type de public de protocoles tels que OAuth 1.0a/2.0, OpenID 2.0 et OpenID Connect 1.0. Cela dépend principalement des applications utilisant l'implémentation du protocole, raison pour laquelle la valeur « Indéfini » a été utilisée pour ces protocoles. A titre d'exemple Facebook Connect est une implémentation d'OAuth 2.0 et s'adresse à une tranche d'âge précise mais est accessible par le « Tout public ». On aurait pu reprendre SAML 2.0 au même titre que la majorité des autres protocoles sous le terme « Indéfini », mais ses fonctionnalités s'adressent plus souvent au « Cercle de confiance » et donc à un panel d'utilisateurs provenant du monde du travail.

Les données que Facebook Connect détient au sujet de la vie privée d'un utilisateur sont énormes. Bien que le format des autorisations soit clair, l'utilisateur est malheureusement rarement éduqué à faire attention à ce type de boites de dialogue. La sensibilisation des utilisateurs est tout autant difficile. Ces derniers points permettent à Facebook Connect de ne pas recevoir la plus mauvaise note pour le critère de respect de la vie privée. Cependant et au vu des résultats obtenus d'un point de vue de la sécurité à la Table 5, le risque d'atteinte à la vie privée est important et ce même lorsqu'un utilisateur y prend garde.

[1] flux des identifiants par mot de passe du propriétaire de la ressource uniquement

[2] flux coté serveur

Les autres protocoles ne sont que très peu décrits dans ce tableau car cette étude de la vie privée est difficilement réalisable sur les aspects théoriques des spécifications. Elle l'est plus facilement dans le cadre d'une implémentation.

Notons tout de même que OAuth 1.0a, OAuth 2.0 et OpenID Connect sont principalement prévus pour déléguer des autorisations et devraient donc logiquement être choisis prioritairement dans le cadre d'un module d'authentification respectueux de la vie privée.

8.2.7 Interopérabilité

La Table 7 donne un aperçu des passerelles théoriques entre les différents modules d'authentification unique. Ce tableau représente donc la façon avec laquelle un système peut facilement s'interconnecter avec un autre. Cette analyse est faite la plupart du temps sur base de la présence d'une extension ou d'une spécification complémentaire permettant d'effectuer cette liaison.

Le symbole «  » sera indiqué au croisement des deux modules concernés par l'interopérabilité. S'il existe une extension, un module particulier pour effectuer l'intégration, il sera indiqué au sein de la même case entre parenthèses. Le nombre de symboles détermine, quant à lui, la difficulté de l'implémentation de cette interopérabilité comme suit :

- un symbole signifie que le couplage est possible mais qu'il nécessite un effort considérable pour le mettre en place et qu'il y a donc très peu de documentations et autres spécifications permettant de le faire ;
- deux symboles signifient qu'il est possible de coupler le module à un autre à l'aide d'une (ou plusieurs) spécification(s). Il s'agit souvent d'une extension précise ;
- trois symboles signifient que cette fonctionnalité est comprise au sein même du protocole ou d'un autre protocole.

A l'inverse, si la compatibilité n'est clairement pas possible et donc qu'il n'y a aucune spécification ou documentation officielle, le symbole «  » sera indiqué.

Les cases résultant de l'interaction avec son homologue sont grisées car le couplage n'est pas d'application (un module ne se couple pas à lui-même). La partie supérieure droite du tableau est également grisée étant donné que les résultats sont identiques à ceux repris dans la partie inférieure gauche.

Le critère appelé **autres** représente la possibilité théorique de coupler le module à l'aide d'un flux étendu du protocole. Il s'agit d'une copie du critère flux d'extension de la Table 4 adapté aux règles en application pour la Table 7.

Finalement le dernier critère intitulé **interopérabilité** est la représentation de la **moyenne** des résultats obtenus par module.

Table 7 : Interopérabilité

	OAuth 1.0a	OAuth 2.0	Facebook Connect	OpenID 2.0	OpenID Connect 1.0	SAML 2.0
OAuth 1.0a						
OAuth 2.0	✗					
Facebook Connect	✗	✓✓✓ Implementation of				
OpenID 2.0	✓✓ (OpenID / OAuth hybrid)	✓✓✓ (OpenID Connect 1.0)	✓✓✓ Via comptes liés de Facebook			
OpenID Connect 1.0	✗	✓✓✓ built-in	✗	✓✓✓ built-in		
SAML 2.0	✗	✓✓ (SAML 2.0 Bearer Assertion Profile for OAuth 2.0)	✗	✓	✓✓ (SAML 2.0 Bearer Assertion Profile for OAuth 2.0)	
Autres	✗	✓✓	✗	✗	✓✓✓	✓✓
Moyenne	0,33	2,17	1	2	1,83	1,17
Interopérabilité	✓	✓✓	✓	✓✓	✓✓	✓

D'un point de vue technique il est toujours possible de créer des liens entre les différents modules. Ce tableau tente surtout de voir quels sont les moyens et documents disponibles pour pouvoir créer ces liens en minimisant les efforts.

L'interconnexion d'OAuth 1.0a à d'autres modules est très peu documentée, raison pour laquelle il reçoit de piètres résultats en la matière. SAML et OAuth 2.0 sont par contre des références dans ce domaine. OpenID Connect étant construit au-dessus de la pile OAuth 2.0, il n'aura aucun mal à se coupler également avec d'autres modules.

Les raisons de ces couplages peuvent être multiples (coupler plusieurs fonctionnalités de modules,…). Un exemple typique d'un tel couplage est l'intégration de PayPal dans SAML par l'intermédiaire d'OpenID. PayPal traitant les informations des cartes bancaires, on garde l'aspect cercle de confiance en l'intégrant avec SAML.

Il n'y a cependant que très peu de documentation pour le faire. Il représente donc un effort considérable pour le mettre en place si l'on souhaite garder un système fiable et respectueux de la vie privé.

Facebook Connect, quant à lui est une implémentation d'OAuth 2.0. Facebook a, pour cette raison, tous les éléments en main pour pouvoir le rendre interopérable avec d'autres modules. A l'heure actuelle, il est possible d'interconnecter Facebook uniquement avec OpenID.

En conclusion, on constate que OAuth 2.0 est maître en matière d'interopérabilité suivi de près par OpenID 2.0 et OpenID Connect 1.0. Rappelons qu'il s'agit de l'interconnexion des modules étudiés dans le cadre de ce mémoire principalement.

8.2.8 Facilité d'implémentation

La Table 8 mesure la difficulté d'implémentation de chacun des protocoles. Pour y arriver, les critères suivants ont été abordés :

- **la disponibilité de frameworks :** quantité de frameworks disponibles et plus précisément leur diversité et leur disponibilité dans différents langages de programmation ;
- **la facilité de la mise en place d'un fournisseur d'identité :** mesure, comme son nom l'indique, la facilité avec laquelle un fournisseur d'identité peut être mis en place ;
- **la facilité de la mise en place d'un fournisseur de services :** mesure, comme son nom l'indique, la facilité avec laquelle un fournisseur de services peut être mis en place ;
- **la qualité globale de la documentation :** il s'agit d'une copie exacte du critère **qualité de la documentation** de la Table 1. La documentation a, en effet, un lien étroit avec l'effort à fournir pour l'implémentation. Une documentation peu claire rend évidemment la tache plus complexe.

Finalement le critère **facilité d'implémentation** donnera une indication générale, moyenne des critères précédemment définis, concernant la complexité de l'implémentation.

Table 8 : Facilité d'implémentation

	OAuth 1.0a	OAuth 2.0	Facebook Connect	OpenID 2.0	OpenID Connect 1.0	SAML
Disponibilité des framework	+++[1]	+[2]	++[3]	+++[4]	N/A	
Qualité de la documentation	+-	++	++	++	+++	++-
Facilité de la mise en place d'un fournisseur d'identité	+++	+++	N/A	+	+	+
Facilité de l'implémentation d'un fournisseur de services	++	+++	+++	+++	++	++
Moyenne	2,38	2,25	2,33	2,25	2	1,38
Facilité d'implémentation	+++-	+++-	+++-	+++-	++	+-

Il n'y a aucun module sortant du lot avec le cote maximale. Ceci reflète bien que mettre en place un module d'authentification unique n'est pas à la portée de tous.

Bien qu'ils soient pour la plupart d'une difficulté d'installation plus ou moins équivalente, les systèmes ont obtenus des cotes parfois très différentes.

La mise en place d'un fournisseur de services par l'intermédiaire de OAuth 2.0, Facebook Connect et OpenID 2.0 est assez simple. Le test a d'ailleurs été effectué avec les deux derniers dans le cadre de ce mémoire.

[1] http://oauth.googlecode.com/svn/code/
[2] http://oauth.net/2/ : section servers et clients
[3] http://developers.facebook.com/docs/sdks/
[4] http://openid.net/developers/libraries/

SAML, à l'inverse demande des efforts plutôt considérables en matière d'implémentation. Son importante documentation et les moyens techniques à mettre en place sont en effet considérables.

OpenID Connect étant encore en phase brouillon et trop peu implémenté, il est difficile de trouver des frameworks lui correspondant à l'heure actuelle. Cette mesure n'a donc pas été prise en considérations dans la moyenne.

Gardons à l'esprit que ce tableau est une moyenne générale dépendant également des services offerts par les protocoles repris à la Table 3. Cette valeur a également été prise en compte et certainement pour la facilité de la mise en place d'un fournisseur d'identité.

8.2.9 Maturité

La Table 9 étudie la maturité des systèmes. Elle se base principalement sur deux critères suivants :

- le **type de document** : l'information est identique au premier bloc de la Table 1 converti en une notion de maturité. La conversion a été faite comme suit: brouillon(⊕), norme et implémentation (⊕⊕), standard (⊕⊕⊕). Un standard étant évidemment plus mature qu'une spécification en phase brouillon ;
- **précédentes versions et apports de protocoles divers :** ce critère permet de mesurer l'expérience qui a été tirée du passé. Voici donc la signification de chacune de ces cotes :
  - un symbole : première spécification en la matière ;
  - deux symboles : une précédente version du protocole existait déjà ;
  - trois symboles : le document est tiré de plusieurs spécifications matures.

  A titre d'exemple, le protocole OAuth 2.0 a été conçu suite aux remarques et manquements du protocole en version 1.0. Il se verra donc attribué deux points.

La **maturité** d'un système est représentée sous le critère de même nom. Il s'agit d'une **moyenne** sur base des deux critères décrits précédemment.

Table 9 : Maturité

	OAuth 1.0a	OAuth 2.0	Facebook Connect	OpenID 2.0	OpenID Connect 1.0	SAML
Type de document	⊕⊕	⊕	⊕⊕	⊕⊕	⊕	⊕⊕⊕
Précédentes versions et apports de protocoles divers	⊕	⊕⊕	⊕⊕	⊕⊕	⊕⊕⊕	⊕⊕⊕
Moyenne	1,5	1,5	2	2	2	3
Maturité	⊕◐	⊕◐	⊕⊕	⊕⊕	⊕⊕	⊕⊕⊕

SAML est sans aucun doute un protocole mature. Il est utilisé dans plusieurs sociétés, diverses institutions de l'Etat,…

OpenID Connect est, pour l'heure actuelle, sanctionné car il n'est qu'en version brouillon actuellement mais devrait faire l'objet d'une standardisation prochainement.

OpenID est un protocole qui a déjà fait beaucoup de chemin grâce à sa première version. Il est à l'heure actuelle un protocole mature dans lequel on peut avoir confiance.

Pour les autres protocoles il s'agit évidemment de protocoles matures mais qui n'ont pas encore totalement fait leurs preuves à l'heure d'écrire ces lignes. Notons que OAuth 1.0a n'évoluera plus en termes de maturité puisqu'il sera, sans doute, abandonné au profit de sa version 2.

8.3 Tableau récapitulatif

La Table 10 est un aperçu global des protocoles qui reprend, point par point, les différentes analyses effectuées tout au long de ce chapitre. Les objectifs de la composition de ce tableau sont, d'une part d'offrir un aperçu rapide des éléments importants de chaque protocole et d'autre part, de créer un outil de travail pour le choix d'un système d'authentification unique.

Table 10 : Aperçu global des systèmes

	OAuth 1.0a	OAuth 2.0	Facebook Connect	OpenID 2.0	OpenID Connect 1.0	SAML
Qualité de la documentation	⊕⊖	⊕⊕	⊕⊕	⊕⊕	⊕⊕⊕	⊕⊕⊖
Type de document	Norme	Brouillon	Implémentation	Norme	Brouillon	Standard
Type de public	Indéfini	Indéfini	Tout public	Indéfini	Indéfini	Cercle de confiance
Richesse des données	⊕⊕	⊕⊕	⊕⊕⊕	⊕⊕	⊕⊕	⊕⊕⊕
Fonctionnalités et services	✓	✓	✓◔	✓✓	✓✓◔	✓✓✓
Scénarios couverts	✓	✓✓◔	◔	✓	✓✓✓	✓✓✓
Risque d'atteinte à la vie privée	N/A	N/A	⊖⊖⊖	N/A	N/A	N/A
Contre-mesures face aux menaces	⛨◔	⛨⛨◔	◔	⛨⛨	⛨⛨⛨	⛨⛨⛨
Interopérabilité	◔	✓✓	✓	✓✓	✓✓	✓
Facilité d'implémentation	⊕⊕⊖	⊕⊕⊖	⊕⊕⊖	⊕⊕⊖	⊕⊕	⊕⊖
Maturité	⊕⊖	⊕⊖	⊕⊕	⊕⊕	⊕⊕	⊕⊕⊕

Il est difficile de pouvoir tirer une conclusion de ce tableau. Certes, il offre un aperçu global, mais il est possible de l'aborder sous différents angles. Chaque système a ses propres points forts, ses points faibles et ses particularités.

De plus, tous les critères ne sont pas toujours très représentatifs de la réalité et nécessite une étude plus approfondie au cas par cas. Le critère « Fonctionnalités », en est un bon exemple. Un utilisateur a rarement besoin de toutes les fonctionnalités proposées par un protocole, il recherche celles dont il a besoin pour son implémentation, les autres étant superflues.

Les besoins d'un utilisateur varient donc suivant les services qu'il cherche à offrir, les données qu'il traite, ses exigences fonctionnelles,… L'élément central pour la navigation dans ce tableau et la sélection est d'ailleurs ce besoin de l'utilisateur. Que compte-t-il faire ? Quel est son but ?

Les critères à prendre en compte pour une implémentation particulière sont difficiles à refléter dans un tel tableau. Tout dépend du but recherché. La sélection ne peut donc se faire qu'au cas par cas.

8.4 Choix d'un système

Au vu du nombre important de critères à prendre en compte pour la sélection d'un système particulier, un arbre de décision aurait eu une allure combinatoire. Bien que certains critères auraient pu être regroupés, il n'en resterait pas moins un nombre important de feuilles rendant l'arbre difficilement lisible.

Par souci d'efficacité, l'approche réalisée l'a été par l'intermédiaire du formulaire repris ci-dessous et du tableau récapitulatif repris à la Table 10. Rappelons que cette approche ne se veut pas apporter une réponse à tous les cas de figure mais donner une piste de ce vers quoi l'utilisateur devrait se tourner parmi les systèmes étudiés. N'oublions pas qu'il existe d'autres systèmes d'authentification unique n'ayant pas été abordés dans ce travail (tels que ceux présentés dans l'annexe 11.1). Le domaine de l'authentification unique via web est large et les nouveautés y sont fréquentes, il est donc indispensable de faire une analyse de tout nouveau système et/ou implémentation qui verraient le jour dans le futur.

Veuillez sélectionner au maximum 5 critères qui vous semblent les plus importants selon vos besoins dans la Table 11 *et leur donner une valeur de 1 à 5. Ces valeurs représentent l'importance que vous accordez à chacun des critères. La valeur « 5 » sera donc attribuée au(x) critère(s) qui selon vous est (sont) le(s) plus important(s).*

Table 11 : Sélection et pondération des critères

Qualité de la documentation	*Richesse des données*	*Fonctionnalités*	*Scénarios*	*Risque d'atteinte à la vie privée*	*Sécurité générale*	*Interopérabilité avec d'autres*	*Facilité d'implémentation*	*Maturité générale*

Une fois cette étape achevée, complétez la Table 12 *à l'aide de la* Table 10 *et du système de cotation associé suivant :*

- *un symbole de couleur verte représente un point (= 1) ;*
- *un symbole «* ⛔ *» représente un point négatif (= -1) ;*
- *un symbole « **N/A** » doit être considéré comme une cote nulle (= 0).*

Chaque cellule de la Table 10 *se voit donc, par le système de cotation ci-dessus, attribuer une cote de -3 à 3. L'application du système de cotation ne doit être faite que pour les critères sélectionnés à la* Table 11*. La cote obtenue doit ensuite être multipliée par la valeur correspondante reprise dans la* Table 11*.*

Table 12 : Application des critères pondérés aux systèmes

Critères	*OAuth 1.0a*	*OAuth 2.0*	*Facebook Connect*	*OpenID 2.0*	*OpenID Connect 1.0*	*SAML*
Total						

Il ne vous reste plus qu'à sommer les différents critères afin de calculer le total de points obtenus par module pour votre cas particulier.

Vous constaterez que le formulaire ne reprend que les critères quantifiables de la Table 10. Les critères textuels sont uniquement présents à titre indicatif pour l'utilisateur. Ils sont par ailleurs repris dans d'autres critères quantifiables (à titre d'exemple, le « type de document » a été repris dans la mesure de la maturité).

Notons que chaque critère devrait être approfondi par les tableaux correspondants afin de mieux cibler le besoin en cas d'égalité mais également pour attribuer une cotation plus précise que la cote globale donnée par défaut dans la Table 10. L'utilisateur pourra donc adapter le résultat qu'il a obtenu. Un exemple très parlant est celui concernant les fonctionnalités. Un utilisateur ne souhaitant effectuer que de la délégation d'autorisations devrait privilégier OAuth 2.0. La cote reprise dans le tableau ne lui attribue que 1,5 point (cf. système de cotation repris dans le formulaire en annexe) alors que d'autres systèmes tels que OpenID et SAML ont obtenu le double des points. Cependant si on approfondit le résultat repris dans la Table 10 par la table correspondante (dans ce cas la Table 3), l'utilisateur devrait adapter la cote du protocole par défaut pour lui donner une cote plus élevée puisqu'il ne s'intéresse pas aux autres fonctionnalités présentes dans le tableau.

L'approche proposée dans ce mémoire est donc la suivante. L'utilisateur complète et suit les recommandations du formulaire proposé ci-dessus sur base de ses exigences et de ses priorités. Une fois ce formulaire complété, il obtient la Table 12. Cette table, si la méthodologie est correctement appliquée avec les différents conseils de cette section, cible les systèmes applicables à votre situation. Le protocole ayant obtenu la note la plus élevée se révèle alors être celui le plus adapté aux besoins de l'utilisateur.

8.5 Synthèse

Les auteurs des spécifications et autres modules étudiés dans ce mémoire ont bien conscience qu'il faut diversifier l'offre pour attirer un maximum de fournisseurs de services et d'identité. Tous les moyens sont bons pour apporter cette diversification et appâter les développeurs à mettre en place un module particulier: une documentation soignée et complète, une offre de scénarios variée, des modules sécurisés et soucieux du respect de la vie privée, l'interopérabilité possible avec d'autres systèmes, toutes sortes de fonctionnalités diverses,… Ces moyens sont principalement les critères repris dans l'aperçu global.

Le critère fonctionnalités et services est souvent un critère décisif pour le choix d'un module. C'est la raison pour laquelle certains systèmes misent énormément sur cet aspect en proposant des services intéressants et variés tels que le service de découverte, la fédération d'identité, la possibilité d'utiliser différents profils et de sélectionner celui adéquat pour visiter un site particulier,… On constatera que très peu de spécifications abordent l'aspect déconnexion unique. SAML le fait et approfondit également la notion de fédération d'identité reprenant les aspects de création d'une identité fédérée et de sa suppression, là où certains protocoles se limitent à la gestion de sessions de l'utilisateur et où d'autres n'abordent même pas l'aspect de la fédération d'identité. Tous ces services peuvent avoir également des répercussions sur le respect de la vie privée et sur la sécurité.

Bon nombre des spécifications étudiées sont des documents fiables en matière de sécurité. Il est indispensable de comprendre et d'implémenter toutes les recommandations d'une spécification qu'elles soient facultatives ou obligatoires.

Malheureusement leur compréhension n'est pas toujours aussi simple et certaines d'entre elles laissent place à toutes sortes d'interprétations pour celui qui l'implémente. L'authentification unique n'est donc pas sans danger! Les menaces sont nombreuses et le risque important. Rappelons qu'en cas d'attaque réussie, les pertes s'étaleront à toutes les applications utilisant le système compromis. Il est également plus qu'indispensable de vérifier systématiquement l'identité de la personne avec qui vous

établissez un lien de confiance. Il faut être vigilant à tout type de message et à ce qu'il couvre. Toute sécurité a un coût et les efforts pour la mettre en place sont très souvent importants. N'oublions pas qu'un système jugé sûr et fiable aujourd'hui peut révéler des failles de sécurité demain.

Les risques de sécurité et d'atteinte à la vie privée sont souvent reportés sur l'utilisateur par l'intermédiaire d'autorisations et autres formes de délégation de droits. Bien souvent, l'utilisateur a tendance à s'habituer à ce genre de pratiques et accepte machinalement les différentes autorisations qui lui sont demandées, sans même les lire. L'utilisateur moyen n'est que rarement conscient des possibles implications que ses autorisations pourraient entrainer. Il est donc primordial que l'utilisateur soit prévenu de façon claire, évidente, et facilement compréhensible.

A l'heure actuelle, bien qu'une majorité de protocoles le permet, il manque toujours un fournisseur d'identité de grande notoriété, respectueux de la vie privée qui filtrerait un maximum les informations fournies (stockant que l'utilisateur à plus de 18 ans en lieu et place de stocker sa date d'anniversaire, stockant la ville de l'utilisateur lorsque l'adresse complète est introduite, etc.). Ce genre de fournisseur d'identité a malheureusement du mal à percer car il n'offre que peu de valeur ajoutée pour les fournisseurs de services.

Le tableau récapitulatif, entre autre utilisé dans la méthodologie, parle de lui-même. Il fait ressortir SAML 2.0 et OpenID Connect 1.0 en tant que principaux favoris des différents protocoles étudiés, suivis de près par OAuth 2.0. Ils ne sont cependant pas les références en la matière quel que soit le cas de figure. En effet, la méthodologie a été mise en place pour justement cibler le protocole adéquat en fonction des besoins et c'est en cela que réside son intérêt.

9 Conclusion

Ce mémoire a plongé le lecteur dans les différents concepts du domaine de l'authentification unique afin d'en éclaircir les termes et les concepts, et d'approfondir certains protocoles et standards dans l'optique de bien cerner les différentes subtilités cachées derrière chacun de ceux-ci.

Chaque système a donc été étudié en détail, la plupart par une approche de type taxonomique avec, entre autres, une attention toute particulière aux questions de respect de la vie privée et de la sécurité. En effet, certains systèmes ne sont pas toujours bien accueillis par les utilisateurs, ils nécessitent d'apporter des preuves de respect de la vie privée et des preuves du bon fonctionnement des mécanismes de sécurité mis en place. Il a donc fallu comprendre, décortiquer chacune des spécifications, les analyser, en faire une implémentation quand cela était envisageable afin de bien en comprendre le fonctionnement. Plusieurs diagrammes de séquence ont été créés afin d'éclaircir ces spécifications et d'en donner un aperçu global plus clair que ceux présents dans certaines spécifications. Facebook Connect est le seul module ayant pu faire l'objet d'une étude approfondie concernant la sécurité et le respect de la vie privée pour la simple et bonne raison qu'il s'agit d'une implémentation d'un protocole. Pour les autres spécifications des protocoles et standards, celles-ci ne font que sensibiliser et donner les recommandations pour l'implémentation.

Une analyse approfondie de ces systèmes selon plusieurs critères a également été réalisée. Ces tableaux ont permis de constater ce que chacun des modules offre sous différents angles. De cette analyse, il est ressorti un tableau récapitulatif. Ce tableau représente un aperçu général des différents modules étudiés les uns vis-à-vis des autres. A partir de ce tableau, une méthodologie de sélection du module d'authentification unique au cas par cas a été développée. Cette approche est d'autant plus intéressante qu'elle laisse la possibilité à l'utilisateur d'inclure dès le début de la sélection, ses propres besoins. Cette technique permet donc de cibler de façon plus précise le système dont l'utilisateur a besoin, chaque module ayant ses avantages et inconvénients.

Une bonne implémentation d'une spécification est une implémentation d'un protocole qui se veut à la fois suffisamment sécurisée, simple, extensible permettant de fournir un minimum d'informations sur l'utilisateur afin d'améliorer son expérience tout en laissant un contrôle complet des informations divulguées par celui-ci.

De tous les protocoles étudiés, OpenID Connect semble rencontrer ces différents critères et paraît être le candidat idéal, sortant du lot des systèmes agissant ailleurs que dans un cercle de confiance. Il reste toutefois difficile de tirer des conclusions concernant OpenID Connect au vu de son stade actuel. A l'inverse, SAML reste le pionner dans un environnement de confiance. Son implémentation n'est cependant pas une sinécure.

La délégation par autorisation de OAuth, la délégation de l'authentification par OpenID, les différents types d'assertions SAML,… Toutes ces méthodes, étudiées au sein de ce mémoire, sont appropriées dans des cas particuliers et toutes peuvent effectuer de l'authentification unique dans un environnement web. Il s'agit donc d'une jungle de protocoles pour une même réalité : l'authentification unique avec des méthodes hétéroclites. Et cette jungle va sans doute continuer de grandir : l'avenir nous réserve certainement d'autres protocoles, qu'il faudra prendre en compte. Le tableau de comparaison repris dans ce mémoire est donc amené à évoluer en fonction de l'offre des protocoles, des standards et des implémentations qui verront le jour dans les années à venir. Le principal est de ne jamais perdre de vue le point central de la méthode de sélection proposée dans ce mémoire : l'authentification est un outil mis à disposition des créateurs de sites web. La sélection de l'une ou l'autre méthode d'authentification doit partir des besoins des développeurs.

10 Bibliographie

Berners-Lee T., Fielding R. & Masinter L., « Uniform Resource Identifier (URI) : Generic Syntax », http://tools.ietf.org/html/rfc3986, janvier 2005 (Date d'accès : 24/12/2011).

Bertino E. & Takahashi K., *Identity Management Concepts, Technologies, and Systems,* ArtTech House, Boston, London 2011.

Bortzmeyer S., « Gestion d'identité avec OpenID », http://2007.jres.org/planning/slides/13.pdf, novembre 2007 (Date d'accès : 20/01/2012).

Calore M., « New 'OpenID Connect' Proposal Could Solve Many of the Social Web's Woes », http://www.webmonkey.com/2010/05/new-openid-connect-proposal-could-solve-many-of-the-social-webs-woes/, 15 mai 2010 (Date d'accès : 12/03/2012).

Cameron K., « 7 Laws of identity », http://www.identityblog.com/?p=1065 , 20 aout 2009 (Date d'accès : 27/02/2012).

Cantor S., Hirsch F., Kemp J. Philpott R., Maler E., « Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0 », http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf, 15 mars 2005 (Date d'accès : 26/04/2012).

Colin J.-N., *Sécurité et fiabilité des systèmes informatiques*, FUNDP, Namur, 2010.

Crockford D., « The application/json Media Type for JavaScript Object Notation (JSON) », http://www.ietf.org/rfc/rfc4627.txt , juillet 2006 (Date d'accès : 28/12/2011).

Detout F., "Deux braqueurs belges trahis par Facebook et un… harpon", http://www.lavenir.net/article/detail.aspx?articleid=DMF20120109_00102837&utm_source=lavenir&utm_medium=newsletter&utm_campaign=daily&utm_content=general-news&ID=164602&POSTALCODE=5100, lundi 09 janvier 2012 (Date d'accès : 12/01/2012).

Dingle P., « OpenID Connect : New, groovy and full of promise », https://www.pingidentity.com/blogs/pingtalk/index.cfm/2011/8/15/OpenID-Connect-New-and-Groovy, 15 aout 2011 (Date d'accès : 10/03/2012).

Dratwa S., "Facebook Connect : le plus grand vol d'informations de l'histoire du web", http://www.lepost.fr/article/2010/05/30/2093099_Facebook-connect-le-plus-grand-vol-d-informations-de-l-histoire-du-web.html, 30 mai 2010 (Date d'accès : 27/11/2011).

Eastlake D., Reagle J. & Solo D., « XML-Signature Syntax and Processing », http://www.ietf.org/rfc/rfc3275.txt, mars 2002 (Date d'accès : 21/02/2012).

EGILIA, « La fédération d'identités (Single Sign On) ou le serpent de mer de la décennie », http://www.egilia.com/fr/articles/20976-la-federation-identites-single-sign-on-le-serpent-mer-la-decennie.html, 2010 (Date d'accès : 14/04/2012).

Facebook, « Core Concepts : Legacy Connect Auth», http://developers.Facebook.com/docs/authentication/connect_auth/, 2011 (Date d'accès : 15/11/2011).

Facebook, « Core Concepts : Authentication », http://developers.Facebook.com/docs/authentication/, 2011 (Date d'accès : 15/11/2011).

Facebook, « Core Concepts : Graph API», http://developers.Facebook.com/docs/reference/api, 2011 (Date d'accès : 15/11/2011).

Facebook, « Se connecter à un autre site à l'aide de Facebook », http://www.facebook.com/about/privacy/your-info-on-other#anothersite, 2011 (Date d'accès : 15/11/2011).

Facebook, « Informations publiques», http://fr-fr.facebook.com/about/privacy/your-info#everyoneinfo, 2011 (Date d'accès : 15/11/2011).

Facebook, « Contrôle de ce qui est communiqué lorsque des personnes qui ont accès à vos informations utilisent des applications», http://www.facebook.com/about/privacy/your-info-on-other#friendsapps, 2011 (Date d'accès : 15/11/2011).

Facebook, « Les informations que nous recevons et leur utilisation», http://www.Facebook.com/about/privacy/your-info, 2011 (Date d'accès : 15/11/2011).

Facebook, « Utilisation des informations que nous recevons», http://www.facebook.com/about/privacy/your-info#howweuse, 2011 (Date d'accès : 15/11/2011).

Ferg B., Fitzpatrick B., Howells C., Recordon D., Hardt D., Reed D., Granqvist H., Ernst J., Bufu J., Hoyt J., Turner K., Scurtescu M., Atkins M. & Glover M., "OpenID Authentication 2.0 – Final", http://openid.net/specs/openid-authentication-2_0.html, 5 Décembre 2007 (Date d'accès : 15/01/2012).

Fiat C., « LinkedIn API avec PHP Zend OAuth», http://www.formatix.eu/wp-content/uploads/2009/12/diagram-oauth-handshake.png, décembre 2009 (Date d'accès : 26/12/2011).

Fletcher G., Lodderstedt T. & Zeltsan Z., « OAuth Use Cases », http://tools.ietf.org/html/draft-zeltsan-oauth-use-cases-02, 11 juillet 2011 (Date d'accès : 13/11/2011).

Franks J., Hallam-Baker P., Hostetler J., Lawrence S., Leach P., Luotonen A. & Stewart L., « http Authentication : Basic and Digest Access Authentication », http://www.ietf.org/rfc/rfc2617.txt , juin 1999 (Date d'accès : 26/12/2011).

Giorgetti S. & Ducamp C., « OpenID, Pourquoi vous devez l'adopter », http://nicolas.barcet.com/drupal/files/active/0/OpenID%20-%20Pourquoi%20l'Adopter.pdf, 30 janvier 2008 (Date d'accès : 22/02/2012).

Hammer E., « Explaining the OAuth Session Fixation Attack », hueniverse.com, http://hueniverse.com/2009/04/explaining-the-oauth-session-fixation-attack/, 23 avril 2009 (Date d'accès : 25/12/2011).

Hammer E., « The OAuth 1.0 Protocol », Internet Engineering Task Force (IETF), http://tools.ietf.org/html/rfc5849, avril 2010 (Date d'accès : 09/10/2011).

Hammer E., « HTTP Authentication : MAC Access Authentication », http://tools.ietf.org/html/draft-ietf-oauth-v2-http-mac-01, 8 février 2012 (Date d'accès : 13/11/2011).

Hammer E., Recordon D. & Hardt D., « The OAuth 2.0 Authorization Protocol - draft-ietf-oauth-v2-22 », http://tools.ietf.org/html/draft-ietf-oauth-v2-22, 8 mars 2012 (Date d'accès : 13/11/2011).

Hirsch F., Philpott R. & Maler E., « Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0 », http://docs.oasis-open.org/security/saml/v2.0/saml-sec-consider-2.0-os.pdf , 15 mars 2005 (Date d'accès : 28/04_2012).

Hughes J., Cantor S., Hodges J., Hirsch F., Mishra P., Philpott R. & Maler E., « Profiles for the OASIS Security Assertion Markup Language (SAML), http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf, 15 mars 2005 (Date d'accès 27/04/2012).

Ideelabor, « OpenID in Estonia », https://openid.ee/en/, 2011 (Date d'accès : 04/02/2012).

Jones M., « JSON Web Key (JWK) », http://openid.net/specs/draft-jones-json-web-key-03.html, 13 décembre 2011 (Date d'accès : 24/03/2012).

Jones M. & Goland Y., « Simple Web Discovery (SWD) », http://openid.net/specs/draft-jones-simple-web-discovery-02.html, 13 décembre 2011 (Date d'accès : 23/03/2012).

Jones M., Balfanz D., Bradley J., Goland Y., Panzer J., Sakimura N. & Tarjan P., « Java Web Signature (JWS) », http://openid.net/specs/draft-jones-json-web-signature-04.html, 13 décembre 2011 (Date d'accès : 27/03/2012).

Jones M., Balfanz D., Bradley J., Goland Y., Panzer J., Sakimura N. & Tarjan P., "JSON Web Token,", http://tools.ietf.org/html/draft-jones-json-web-token, décembre 2011 (Date d'accès : 20/03/2012).

Jones M., Rescoria E. & Hildebrand J., « JSON Web Encryption (JWE) », http://openid.net/specs/draft-jones-json-web-encryption-02.html, 13 décembre 2011 (Date d'accès : 28/03/2012).

Jones M., Hardt D. & Recordon D., « The OAuth 2.0 Authorization Protocol : Bearer Tokens », http://tools.ietf.org/html/draft-ietf-oauth-v2-bearer-18, 12 mars 2012 (Date d'accès : 13/11/2011).

Journal du net, « OpenID : vulnérabilité détectée mais pas exploitée », http://www.journaldunet.com/solutions/securite/vulnerabilite-dans-openid-0511.shtml, 10 mai 2011 (Date d'accès : 25/02/2012).

Kiani K., « OAuth and OpenID – Securing the Insecure – Hack.lu », http://www.n0secure.org/2011/09/oauth-and-openid-securing-insecure.html, 21 septembre 2011 (Date d'accès : 22/02/2012).

Laurie B. & Clayton R., « OpenID Advisory », http://www.links.org/files/openid-advisory.txt, 08 Aout 2008 (Date d'accès : 20/02/2012).

Lei J., Takabi H. & Joshi J.B.D, "Security and Privacy Risks of Using E-mail Address as an Identity", *SocialCom,* 906-913, 30 septembre 2010.

Lodderstedt T., McGloin M. & Hunt P., « OAuth 2.0 Treat Model and Security Considerations », http://tools.ietf.org/html/draft-ietf-oauth-v2-threatmodel-01 , 26 octobre 2011 (Date d'accès : 13/11/2011).

McCarty B., « Facebook Connect, OAuth and OpenID : The differences and the future », http://thenextweb.com/socialmedia/2010/11/04/facebook-connect-oauth-and-openid-the-differences-and-the-future/, 04 novembre 2010 (Date d'accès : 02/05/2012).

Madsen P., Maler E., « SAML v2.0 Executive overview », http://www.oasis-open.org/committees/download.php/11785/sstc-saml-exec-overview-2.0-draft-06.pdf, 12 avril 2005 (Date d'accès : 21/04/2012).

de Medeiros B., Scurtescu M. & Tarjan P., « OAuth 2.0 Multiple Response Type Enconding Practices – draft 03 », http://openid.bitbucket.org/oauth-v2-multiple-response-types-1_0.html, 23 décembre 2011 (Date d'accès : 09/03/2012).

Miller J., « Yadis Specification Version 1.0 », http://yadis.org/papers/yadis-v1.0.pdf, 18 mars 2006 (Date d'accès : 15/01/2012).

Mishra P., Philpott R. & Maler E., « Conformance Requirements fot the OASIS Security Assertion Markup Language (SAML) V2.0 », http://docs.oasis-open.org/security/saml/v2.0/saml-conformance-2.0-os.pdf, 15 mars 2005 (Date d'accès 23/04/2012).

Moiny J.-P., « Facebook, Gare au clic », *Le Soir,* mercredi 29 avril 2009.

Moiny J.-P., "Facebook au regard des règles européennes concernant la protection des données", *Strada*, paragraphe 18 p246, février 2010.

Moiny J.-P., « Sur Internet, ce que vous ne contrôlez pas est définitivement hors de votre portée », A*thena n°271*, p24-25, mai 2011.

Moonz, « Mozilla concurrence OpenID », http://linuxfr.org/users/moonz/journaux/mozilla-concurrence-openid, 29 Juillet 2011 (Date d'accès : 18/02/2012).

Nam Ko M., Gorrell P. & Shehab M., *« Social-Network Connect Services », ComputingNow,* 37-43, Aout 2010.

Prodromou E., « OpenID et les Soucis de vie privée », http://xtof.livejournal.com/20061.html, 20 janvier 2008 (Date d'accès : 18/02/2012).

Ragouzis N.,Hughes J., Philpott R., Maler E., Madsen P., Scavo T., "Security Assertion Markup Language (SAML) V2.0 Technical Overview", http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf, 25 mars 2008 (Date d'accès: 21/04/2012).

Raynal J., « 600 000 comptes Facebook victimes de tentatives de hacking chaque jour », http://frenchweb.fr/infographie-600-000-comptes-facebook-victimes-de-tentatives-hacking-chaque-jour-50227/, 31 octobre 2011 (Date d'accès : 29/11/2011).

Reed D. & McAlpin D., «Extensible Resource Identifier Syntax v2 .0 », http://www.oasis-open.org/committees/download.php/15377, 14 novembre 2005 (Date d'accès : 28/02/2012).

Rob B., « Facebook, lié à de plus en plus de divorces », http://www.lavenir.net/article/detail.aspx?articleid=DMF20120103_018&utm_source=lavenir&utm_medium=newsletter&utm_campaign=soir&utm_content=general-news, 03 janvier 2012 (Date d'accès : 27/11/2011).

Sakimura N., « OpenID Connect Implementer's Drafts Approved », http://openid.net/2012/02/16/openid-connect-implementers-drafts-approved/, 16 février 2012 (Date d'accès : 09/03/2012).

Sakimura N., Bradley J., Jones M., de Medeiros B., Mortimore C. & Jay E., « OpenID Connect Session Management 1.0 –draft 05 », http://openid.net/specs/openid-connect-session-1_0.html, 23 décembre 2011 (Date d'accès : 09/03/2012).

Sakimura N., Bradley J., Jones M. & Jay E., « OpenID Connect Discovery 1.0 –draft 07 », http://openid.net/specs/openid-connect-discovery-1_0.html, 23 décembre 2011 (Date d'accès : 09/03/2012).

Sakimura N., Bradley J., de Medeiros B., Jones M., Jay E. & Mortimore C., « OpenID Connect Basic Client 1.0 –draft 16 », http://openid.net/specs/openid-connect-basic-1_0.html, 27 février 2012 (Date d'accès : 09/03/2012).

Sakimura N., Bradley J. & Jones M., « OpenID Connect Dynamic Client Registration 1.0 – draft 09 », http://openid.net/specs/openid-connect-registration-1_0.html, 27 février 2012 (Date d'accès : 09/03/2012).

Sakimura N., Bradley J., de Medeiros B., Jones M. & Jay E., « OpenID Connect Standard 1.0 –draft 08 », http://openid.net/specs/openid-connect-standard-1_0.html, 27 février 2012 (Date d'accès : 09/03/2012).

Sakimura N., Recordon D., Bradley J., de Medeiros B., Jones M. & Jay E., « OpenID Connect Messages 1.0 – draft 08 », http://openid.net/specs/openid-connect-messages-1_0.html, 27 février 2012 (Date d'accès : 09/03/2012).

SANS, « Top 20 Internet Security Problems, Threats and Risks », http://www.sans.org/top20/2007/, 2007 (Date d'accès : 12/04/2012)

Schneirer B., « Cryptanalysis of SHA-1 », http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html , 18 février 2005 (Date d'accès : 12/11/2011).

Tamada S., « Facebook Graph API to Post Status Update », http://www.9lessons.info/2011/03/facebook-graph-api-to-post-status.html, 31 Mars 2011 (Date d'accès : 15/11/2011).

The identity corner, « The problem(s) with OpenID », http://www.untrusted.ca/cache/openid.html, 22 aout 2007 (Date d'accès : 24/02/2012).

Trenka T., "Introducing OAuth in DojoX", http://www.sitepen.com/blog/2009/02/19/introducing-oauth-in-dojox/ ,19 février 2009 (Date d'accès : 15/10/2011).

Villacres C., "L'authentification de A à Z", *Ernst & Young LLP - Security & Technology Services*, page 1-7, date d'édition inconnue.

Walther P., "Hauptsemina Web Engineering im WS 2010/2011 – Identity & Authentication", http://vsr.informatik.tu-chemnitz.de/edu/2011/webe-seminar-ss/drafts/03/, mars 2011 (Date d'accès : 26/02/2012).

Wikipedia, « Facebook », http://fr.wikipedia.org/wiki/Facebook, 13 mars 2012 (Date d'accès : 18/03/2012).

Wikipedia, « URI scheme », http://en.wikipedia.org/wiki/URI_scheme#Generic_syntax, 12 janvier 2012 (Date d'accès 17/01/2012).

Zimmer M., "But the data is already public: on the ethics of research in Facebook", http://michaelzimmer.org/2009/06/18/draft-paper-but-the-data-is-already-public/, juin 2009 (Date d'accès: 29/11/2011).

11 Annexes

11.1 Les autres systèmes

Parmi cette jungle, il existe évidemment d'autres implémentations et protocoles d'authentification unique dans un environnement web. Cependant au vu de la limite imposée du nombre de pages, du grand nombre de systèmes d'authentification et de la faible popularité de certains, ces systèmes n'ont pas été présentés en détail mais sont mentionnés à titre indicatif avec un bref résumé de chacun d'eux. Voici donc quelques exemples de systèmes d'authentification unique dans un environnement web qui n'ont pas été abordés dans ce mémoire.

BBAuth (Browser-Based Authentication)
Description Projet de Yahoo abandonné pour laisser place au protocole OpenID-OAuth Hybrid.
Documentation : http://developer.yahoo.com/bbauth/

CAS (Central Authentication Service)
Description: Protocole basé sur l'échange de tickets (ne contenant aucune information) comme le fait Kerberos.
Documentation : http://www.jasig.org/cas/protocol

Google Connect
Description: Google Connect est une implémentation d'OpenID ou OAuth. Il existe, en effet, une implémentation de chacun de ces protocoles.
Documentation : https://developers.google.com/gdata/docs/auth/overview

Liberty Alliance
Description: Ensemble de spécifications publiques rédigées par un consortium d'industriels. Implémentations basées sur des standards ouverts (principalement SAML).
Documentation : http://www.projectliberty.org/liberty/specifications_1

Microsoft LiveID
Description: Solution propriétaire de Microsoft permettant de faire de l'authentification unique.
Documentation : https://accountservices.passport.net/ppnetworkhome.srf

OAth v1 & 2
Description: Collaboration du secteur industriel proposant des standards ouverts pour un ensemble de technologies d'authentification dans le but de réduire les coûts et simplifier leur utilisation.
Documentation : http://www.openauthentication.org/specifications

Shibbolet
Description: Implémentation basée sur des standards ouverts. Elle se base sur SAML pour échanger l'assertion d'authentification et les attributs de l'utilisateur.
Documentation : http://shibboleth.internet2.edu/

11.2 OAuth v1.0a : le flux d'authentification et ses paramètres

La Figure 11-1 est une vue synthétique des différents paramètres échangés aux différentes étapes du flux d'autorisation OAuth 1.0a.

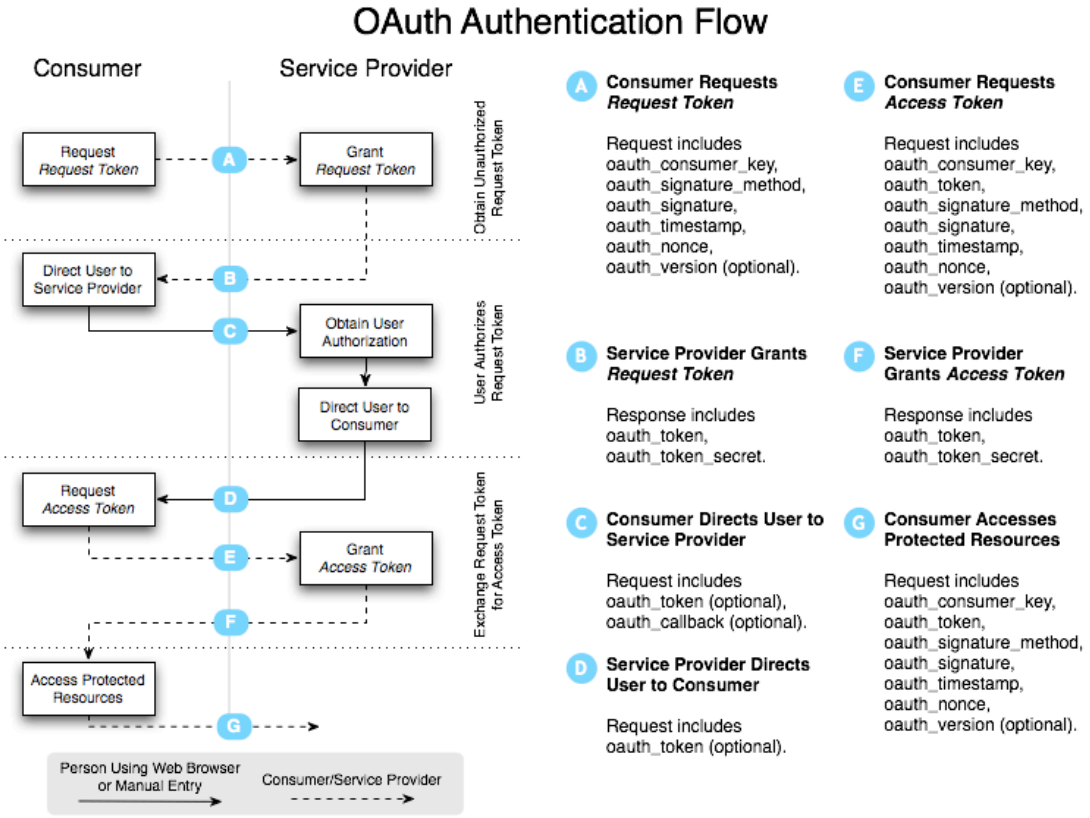


Figure 11-1 : Flux d'authentifcation OAuth v1.0a et les paramètres correspondants [Fiat, 2009]

11.3 Facebook Connect : configuration de l'application Facebook

Lors de l'implémentation du flux « coté serveur » de Facebook Connect, il vous faudra configurer votre application dans Facebook Developer. Pour ce faire, les étapes suivantes sont nécessaires :

- création de son application sur la communauté Facebook Developer et l'acceptation des politiques de la plate-forme Facebook sont représentés à la Figure 11-2 ;

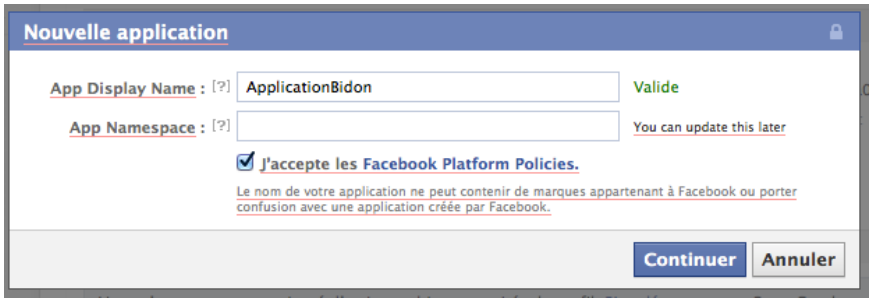


Figure 11-2 : Création d'une nouvelle application sur Facebook Connect

La Figure 11-3 représente l'invite d'insertion du captcha :

Figure 11-3 : Insertion du captcha de sécurité Facebook Connect

- terminer l'inscription de votre application en encodant un code reçu via son GSM (en ayant pris soin au préalable de compléter le numéro de téléphone sur son profil Facebook). Une illustration de ce SMS est reprise à la Figure 11-4 ;

Figure 11-4 : Validation du compte Facebook Developers par SMS

- une fois le code reçu et le mobile validé vous accéderez au panneau de configuration de votre application.
A noter qu'une confirmation du mot de passe représenté à la Figure 11-5 est nécessaire après authentification sur Facebook pour accéder à la partie Developers ;

Figure 11-5 : Sécurité de Facebook Developers

Le panneau de configuration de votre application se présente comme sur la Figure 11-6.

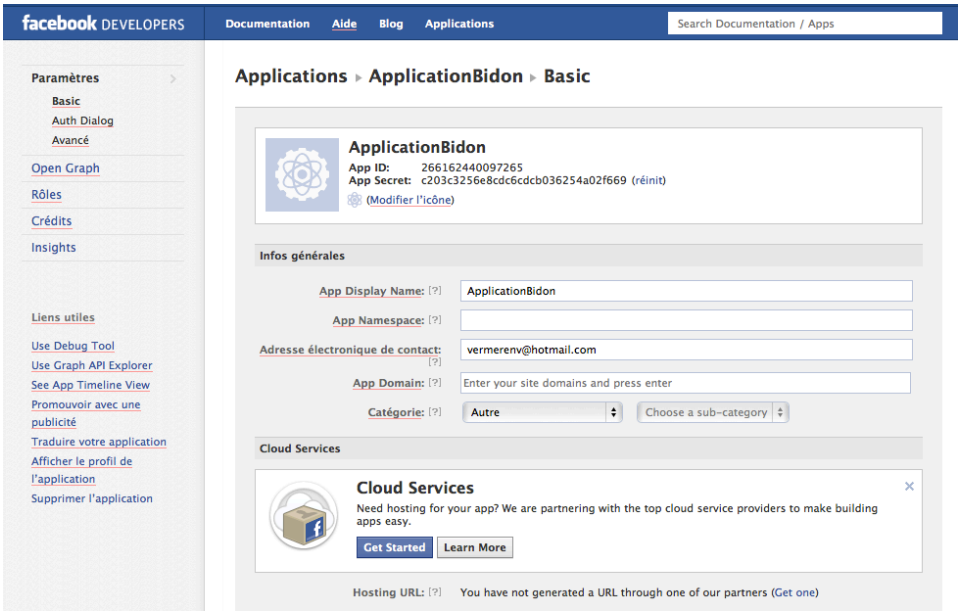


Figure 11-6 : Panneau de configuration Facebook Developers

Ce panneau est divisé en 3 parties nommées :

a. Basic : permet de définir les paramètres nécessaires à l'accès à votre application via Facebook Connect ;
b. Auth Dialog : permet de définir les autorisations qui seront soumises à l'utilisateur concernant son profil et la façon dont seront présentées ces données ;
c. Avancé : permet de définir des paramètres avancés (sécurité, authentification, migration,…) afin de donner la possibilité de personnaliser son site.

11.4 OpenID 2.0 : choisir son fournisseur

Le tableau repris à la Figure 11-7 est une représentation des fonctionnalités de différents fournisseurs OpenID.

	Name	Ease of use	Security	Remembers information	Multiple profiles	Anti-phishing measures	Password protected
	AOL/AIM	3	4				✓
	Blogger	9	4				✓
	ClaimID	9	4	✓			✓
	GetOpenID	7	4				✓
	Google	7	4	✓			✓
	LiveJournal	9	1				✓
★	MyOpenID	8	9	✓	✓	✓	✓
	MySpace	6	1				✓
	Sxipper	7	8	✓	✓	✓	✓
★	VeriSign	7	7	✓		✓	✓
	Vidoop	8	4	✓	✓	✓	
	WordPress	5	1	✓		✓	✓
★	Yahoo!	10	4			✓	✓

Figure 11-7 : Analyse comparative de différents fournisseurs OpenID[1]

Au sein de ce tableau, les fournisseurs d'identité précédé d'un symbole « ★ » sont ceux recommandés par le site http://openidexplained.com/. Les critères du tableau sont relativement explicites et ne nécessitent pas d'explication complémentaire. Notons que les mesures utilisées pour calculer le critère sécurité des fournisseurs sont détaillées sur le site http://openidexplained.com/.

On remarquera que, parmi tous les fournisseurs d'identité représentés dans ce tableau, MyOpenID se démarque globalement de tous les autres grâce aux fonctionnalités et la sécurité qu'il offre mais également par sa facilité d'utilisation.

[1] Ce graphe provient du site http://openidexplained.com/get

11.5 OpenID Connect 1.0 : spécifications complémentaires

11.5.1 Gestion de session

Ce document décrit comment gérer les sessions au sein d'OpenID Connect.

OpenID Connect supporte la gestion de sessions « en cycle de vie » et la synchronisation pour les tiers supportant l'authentification avec un serveur d'authentification. De plus, la gestion des sessions entre 4 parties telles que les applications mobiles, natives ou de bureau utilisant les identifiants du serveur d'autorisation, est également supportée. Notons que la spécification est décrite en termes de endpoints, cela permet entre autre de laisser le choix libre de l'endroit ou l'on souhaite le déployer.

11.5.1.1 Création de sessions

Pour créer une session, le client envoie une requête d'autorisation vers le serveur d'autorisation. En réponse, le jeton ID sera renvoyé avec le jeton d'accès.

Le jeton ID est utilisé comme une clé pour une session authentifiée d'un utilisateur et contient les éléments pour l'utilisateur. Il peut être utilisé pour accéder à l'information de session pour une session authentifiée ou pour transmettre une session à d'autres applications.

Flux d'octroi implicite (user-agent)

Les user-agents peuvent utiliser le flux d'octroi implicite OAuth 2.0. Ce flux peut être représenté comme suit à la Figure 11-8.

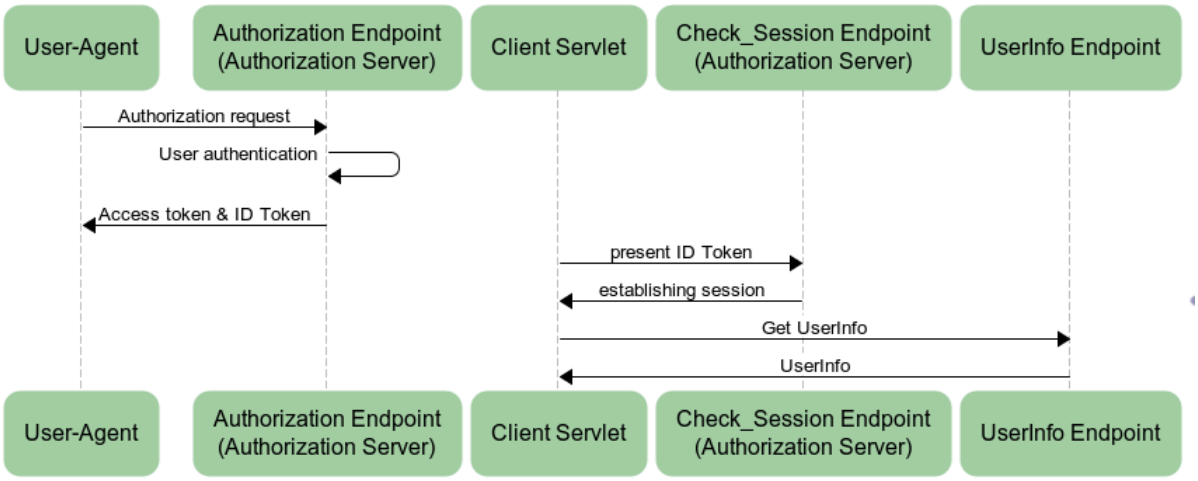


Figure 11-8 : Utilisation du flux d'octroi implicite OpenID Connect 1.0 (jeton ID)

La réponse d'autorisation doit renvoyer les paramètres dans le fragment de la réponse à la requête ainsi que le jeton ID.

Flux par code d'autorisation

Les serveurs web clients peuvent utiliser le flux par code d'autorisation. Le jeton ID est renvoyé avec le jeton d'accès une fois que le client a soumis le code d'autorisation sur l'endpoint du jeton d'accès. Ce flux peut être représenté comme suit à la Figure 11-9.

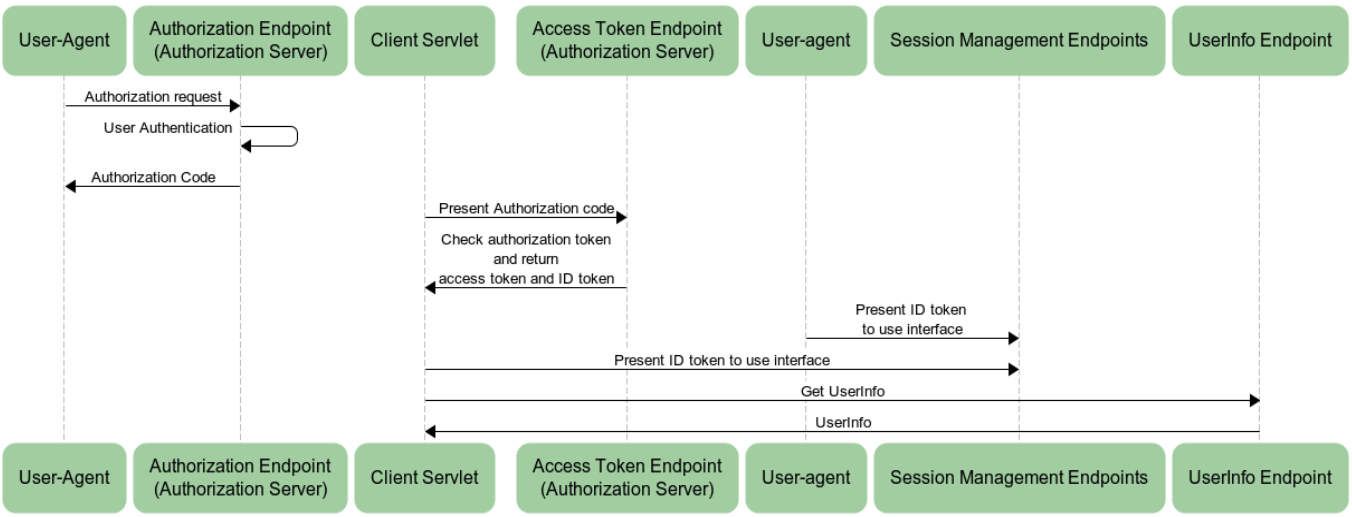


Figure 11-9 : Flux par code d'autorisation OpenID Connect 1.0 (jeton ID)

Les clients utilisent le code d'autorisation pour former la requête à destination du endpoint du jeton afin d'y retirer le jeton d'accès et le jeton ID. Ceux-ci sont renvoyés dans la réponse.

Application native pour un quatrième intervenant

Les acteurs dans les applications natives sont : l'utilisateur, l'application native (de bureau), le serveur d'autorisation et l'application web du servlet client. L'application native utilise des ressources protégées du servlet client mais elle s'intègre directement avec les services d'authentification du serveur d'autorisation. Elle redirige l'utilisateur pour effectuer l'authentification sur le serveur d'autorisation afin d'obtenir les jetons d'accès et le jeton ID. Les jetons peuvent alors être utilisés pour accéder aux ressources protégées sur le servlet client. Le processus pour obtenir un jeton ID pour l'application native est très similaire à celui de l'utilisation de la méthode de flux par code d'autorisation. La cible publique du jeton ID n'est pas l'application native cette fois mais le servlet client. Le flux générique de ce processus est repris à la Figure 11-10.

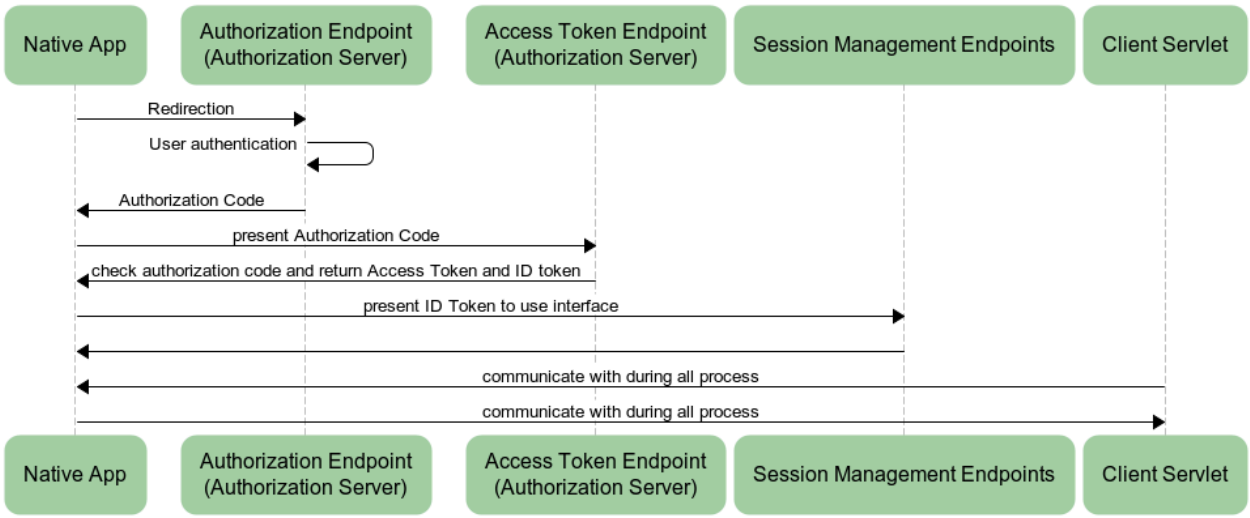


Figure 11-10 : Flux générique OpenID Connect 1.0 : application native avec un 4ème intervenant

Lorsque l'on accède à des ressources protégées par un servlet client, l'application native envoie le jeton ID dans l'en-tête de la requête. Le servlet client peut vérifier la

validité du jeton ID en vérifiant l'information cryptographique ou en envoyant le jeton ID vers le endpoint CheckID.

Chargement du navigateur

Certaines applications natives peuvent souhaiter démarrer une session authentifiée pour le même utilisateur. Pour ce faire, l'application native démarre un navigateur avec la localisation du servlet client tout en lui passant un jeton ID en tant que paramètre de la requête. Le servlet client initialise immédiatement une requête vers le endpoint de rafraichissement de session avec le jeton ID. L'utilisateur peut avoir besoin de s'authentifier à nouveau sur le serveur d'autorisation. Le servlet client obtient alors un jeton ID qui est synchronisé avec la session du serveur d'autorisation.

Le diagramme de séquence de la Figure 11-11 représente ce flux.

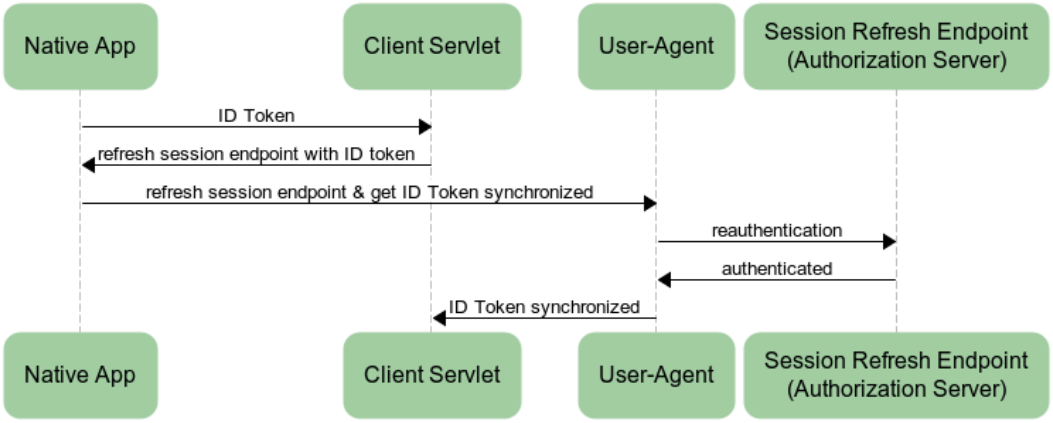


Figure 11-11 : Jeton ID synchronisé entre l'application native et le user-agent

11.5.1.2 Endpoint de gestion de sessions

Pour gérer une session, le client envoi une requête vers les endpoints de gestion de sessions sur le serveur d'autorisation. Les endpoints de gestion de sessions du serveur d'autorisation sont les :

- endpoint de rafraichissement de session: permet de rafraichir un jeton ID expiré ;
- endpoint de terminaison de session: permet de mettre fin à une session.

Le jeton ID

Un jeton ID est utilisé pour lier la session d'un utilisateur connecté avec le serveur d'autorisation, mais dans certains cas, tels que lors des accès hors connexion par un serveur web ou une application native ce n'est pas possible. Les jetons ID obtenus dans les scénarios suivants sont liés à un état « connecté » de l'utilisateur sur le serveur d'autorisation :

- renouveler un code pour un jeton d'accès et un jeton ID par le biais de communications indirectes à travers le navigateur ;
- obtenir un jeton d'accès et un jeton ID dans la réponse d'autorisation à travers le navigateur ;
- obtenir un jeton ID sur le endpoint de rafraichissement de session en soumettant un jeton ID reçu précédemment.

Les jetons ID sont synchronisés avec la session. Si un jeton ID est obtenu en soumettant un jeton de rafraichissement sur le endpoint du jeton d'accès, alors le jeton ID résultant n'est pas lié à l'état « connecté » de l'utilisateur sur le serveur d'autorisation. Le client peut être hors ligne ou l'utilisateur peut être déconnecté du serveur d'autorisation. Si une session liée à un jeton ID est souhaitée, le client devrait

obtenir un nouveau jeton ID en envoyant une requête vers le endpoint de rafraichissement de session.

11.5.2 Standard

Protocole HTTP de liaisons pour les requêtes et réponses aux messages d'OpenID Connect Messages 1.0.

11.5.2.1 Endpoint d'autorisation

Le endpoint d'autorisation rend des services et effectue des demandes d'autorisations à l'utilisateur. Il divulgue des informations aux fournisseurs de services. Lorsqu'un utilisateur accède à une application d'un fournisseur de services qui nécessite l'identifiant de l'utilisateur et d'autres informations, le fournisseur de services l'envoie vers le endpoint d'autorisation afin qu'il s'y authentifie et reçoive les autorisations nécessaires. Le serveur d'autorisation fournit alors un jeton ID affirmant l'identifiant de l'utilisateur et un jeton d'accès permettant au client d'accéder aux informations de l'utilisateur sur les endpoints correspondants. Ceux-ci peuvent effectuer différentes actions ou renvoyer différentes informations sur base des champs d'application (valeurs *scope)* associés avec le jeton d'accès.

Le flux du protocole

Le flux détermine comment le jeton d'accès et le jeton ID sont renvoyés au client. Les jetons d'accès sont des identifiants utilisés pour accéder aux ressources protégées. Ils représentent une autorisation du propriétaire de la ressource et ne doivent pas être exposés aux tiers non autorisés

Les requêtes d'autorisation peuvent suivre deux flux afin d'acquérir les jetons d'accès et les jetons ID :

- le flux implicite : utilisé par les clients implémentés dans un navigateur utilisant des langages de script. Le jeton d'accès et le jeton ID sont renvoyés directement au client, lequel peut les exposer au propriétaire de la ressource et autres applications ayant accès au user-agent. Le serveur d'autorisation n'effectue pas d'authentification client avant de fournir le jeton d'accès ;
- le flux par code d'autorisation : renvoie un code au client, pouvant être échangé contre un jeton d'accès. Cela a pour principal bénéfice de ne pas exposer le jeton d'accès au propriétaire de la ressource et d'autres applications ayant accès au user-agent du propriétaire de la ressource. Le serveur d'autorisation peut aussi authentifier le client avant l'échange du code d'autorisation avec le jeton d'accès.

Le flux par code d'autorisation est préférable pour les clients pouvant maintenir de façon sécurisée un secret client entre eux et le serveur d'autorisation. Dans tous les autres cas, le flux implicite est préférable.

Le flux par code d'autorisation

Les étapes du flux par code d'autorisation sont les suivantes :

1. Le client prépare la requête d'autorisation avec les paramètres désirés.
2. Le client envoie une requête au serveur d'autorisation.
3. Le serveur d'autorisation authentifie l'utilisateur.
4. Le serveur d'autorisation obtient le consentement et l'autorisation de l'utilisateur.
5. Le serveur d'autorisation renvoie l'utilisateur vers le client avec un code d'autorisation.
6. Le client demande une réponse en utilisant le code d'autorisation sur le endpoint du jeton.
7. Le client reçoit une réponse contenant un jeton d'accès et un jeton ID dans le corps de la réponse.

8. (facultatif) Le client valide le jeton ID sur le endpoint CheckID.
9. (facultatif) Le client reçoit la réponse du jeton ID avec l'identifiant de l'utilisateur.
10. (facultatif) Le client accède au endpoint UserInfo avec le jeton d'accès.
11. (facultatif) Le client reçoit la réponse du endpoint UserInfo.

Le flux implicite

Les étapes du flux implicite sont les suivantes :

1. Le client prépare la requête d'autorisation contenant les paramètres désirés.
2. Le client envoie une requête au serveur d'autorisation.
3. Le serveur d'autorisation authentifie l'utilisateur.
4. Le serveur d'autorisation obtient le consentement et l'autorisation de l'utilisateur.
5. Le serveur d'autorisation renvoie l'utilisateur vers le client avec un jeton d'accès et un jeton ID s'il a été demandé.
6. (facultatif) Le client valide le jeton ID sur le endpoint CheckID.
7. (facultatif) Le client reçoit la réponse du jeton ID avec l'identifiant de l'utilisateur.
8. (facultatif) Le client accède au endpoint UserInfo avec le jeton d'accès.
9. (facultatif) Le client reçoit la réponse du endpoint UserInfo.

Quel que soit le flux, le message réceptionné doit être vérifié à chaque étape conformément aux règles de vérification définies dans la spécification OpenID Connect Message 1.0.

La requête d'autorisation

Lorsque l'utilisateur souhaite accéder à une ressource protégée, le fournisseur de services prépare une requête d'autorisation à destination du endpoint d'autorisation.

Il existe trois méthodes pour construire et envoyer la requête vers le endpoint d'autorisation :

- **méthode par simple requête** : utilisée dans de simples cas lorsque les éléments UserInfo et jeton ID sont souhaités ;
- **méthode par requête de paramètre** : utilisée lors de l'envoi d'un « OpenID Request Object » si le client souhaite retirer un ensemble différent d'éléments UserInfo et le jeton ID. Cette méthode permet aussi aux requêtes d'être signées ou chiffrées ;
- **méthode par requête de fichier** : fonctionne de manière similaire à la méthode par requête de paramètre, à la seule différence que la méthode par requête de fichier envoie une URL comme référence à un « OpenID Request Object ». il permet également d'envoyer de grandes requêtes signées et compactées même avec des user-agents limités.

11.5.2.2 Endpoint du jeton

Le endpoint du jeton gère les requêtes servant à retirer ou à rafraichir aussi bien les jetons d'accès que les jetons ID.

Demande d'un jeton d'accès

Afin de retirer un jeton d'accès, le client doit posséder le code d'autorisation.

Pour obtenir un jeton d'accès, un jeton de rafraichissement ou un jeton ID, le client doit s'authentifier sur le endpoint du jeton en utilisant la méthode d'authentification enregistrée pour ce client.

Le serveur d'autorisation doit :

- exiger l'authentification cliente pour tout client fournissant des identifiants client ;
- authentifier le client si l'authentification cliente est comprise et s'assurer que le code d'autorisation a été fourni au client authentifié ;
- vérifier que le code d'autorisation est valide ;
- s'assurer que le paramètre *redirect_uri* est présent si le paramètre *redirect_uri* est inclus dans la requête d'autorisation initiale et que ses valeurs sont identiques.

Dès réception de la requête du jeton, le serveur d'autorisation doit, soit renvoyer :
- une réponse avec le jeton d'accès ;
- une erreur correspondant au code d'autorisation reçu.

Rafraichir un jeton d'accès

Afin de rafraichir un jeton d'accès, le client doit s'authentifier sur le endpoint du jeton en utilisant la méthode d'authentification enregistrée pour ce client.

Le serveur d'autorisation, quant à lui, doit vérifier la validité du jeton de rafraichissement. Dès la réception d'une requête de rafraichissement de jeton, le serveur d'autorisation renvoie :
- soit une réponse correcte contenant le jeton d'accès ;
- soit une erreur correspondant au jeton de rafraichissement reçu.

11.5.2.3 Endpoint CheckID

Le endpoint CheckID valide le jeton ID et renvoie un objet JSON contenant les éléments du jeton ID. Cet endpoint est utilisé par les clients n'étant pas capables ou ne souhaitant pas traiter directement les jetons ID. Dans ce cas, le client peut traiter le jeton ID comme une valeur incompréhensible.

Afin de demander de l'information concernant l'authentification effective de l'utilisateur, une requête est effectuée vers le endpoint CheckID en utilisant le jeton ID comme jeton d'accès. Le client doit s'assurer que le endpoint utilisé est le endpoint CheckID dans lequel on a placé sa confiance.

Le endpoint CheckID doit renvoyer un élément JSON sérialisé associé avec le jeton ID dans le corps de la réponse.

Afin de vérifier la validité de la réponse, le client doit :
- vérifier que le fournisseur OpenID est bien le fournisseur prévu à l'aide d'une vérification du certificat du serveur ;
- suivre les règles de la section 5.4 de la spécification OpenID Connect Messages.

11.5.2.4 Endpoint UserInfo

Afin d'obtenir les éléments demandés au sujet de l'utilisateur, le client effectue une requête vers le endpoint UserInfo.

L'élément *user_id* du endpoint UserInfo doit correspondre exactement à l'élément *user_id* du jeton ID. Si ce n'est pas le cas, les éléments du endpoint UserInfo ne pourront être utilisés.

Dès la réception de la requête UserInfo, le endpoint de même nom doit renvoyer la réponse sérialisée en JSON dans le corps de la réponse.